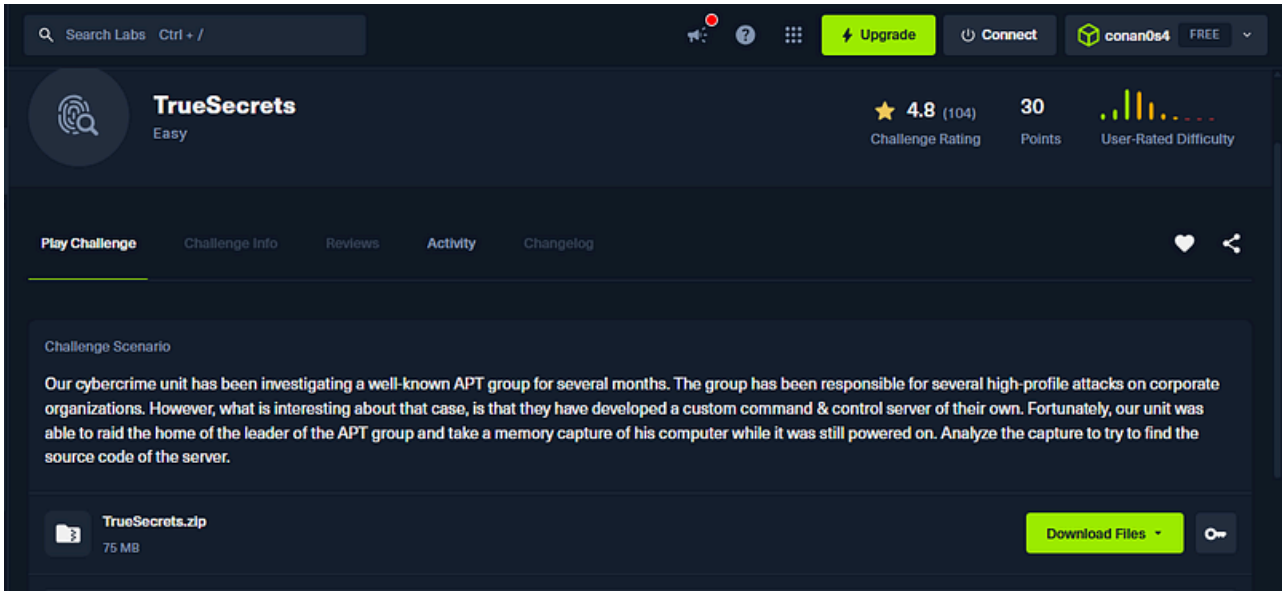
HTB Labs : TrueSecrets |Digital Forenics Write-up



Challenge Scenario

Our cybercrime unit has been investigating a well-known APT group for several months. The group has been responsible for several high-profile attacks on corporate organizations. However, what is interesting about that case, is that they have developed a custom command & control server of their own. Fortunately, our unit was able to raid the home of the leader of the APT group and take a memory capture of his computer while it was still powered on. Analyze the capture to try to find the source code of the server.

Challenge Information

- **Category:** Memory Forensics
- **Memory Image:** TrueSecrets.raw

Tools Used:

- Volatility Workbench v3.0 (GUI)
- Volatility CLI v2.1

What Is Volatility?

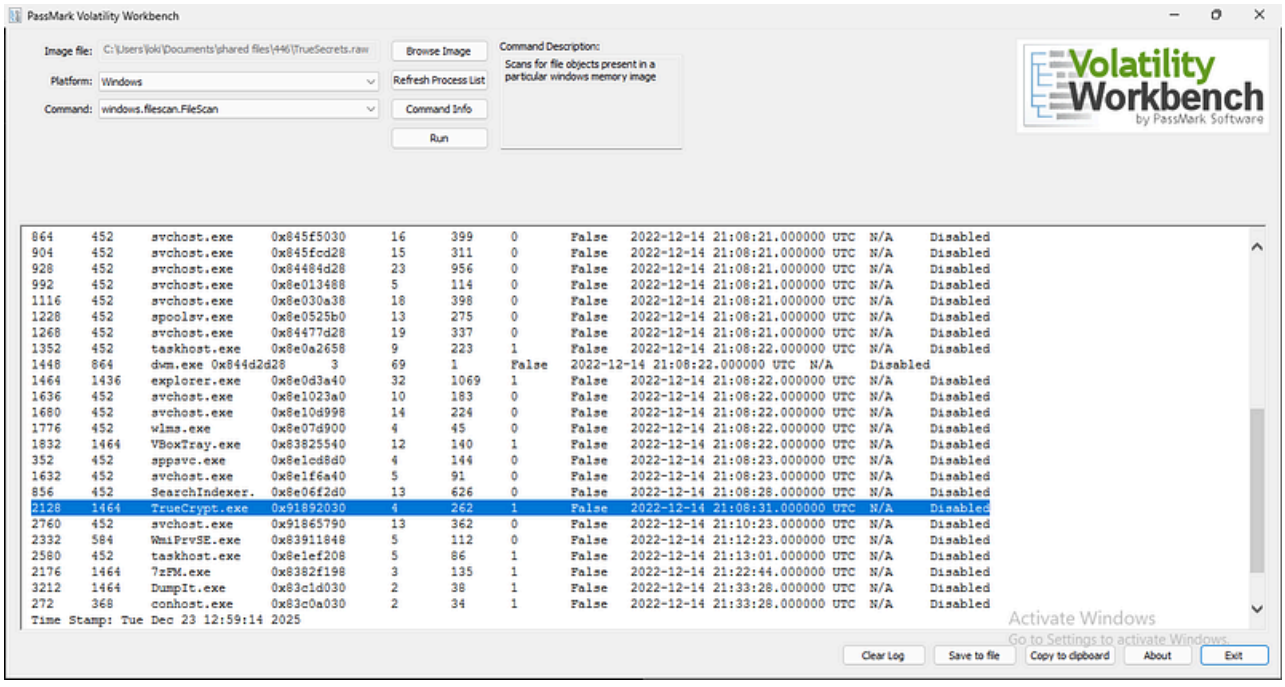
Volatility is an **open-source memory forensics framework** used for **incident response, malware analysis, and digital investigations**. It allows investigators to extract processes, files, credentials, encryption keys, and other forensic artifacts directly from volatile memory (RAM).

Initial Analysis: Process Enumeration

After refreshing the process list, an interesting process stood out:

2128 1464 TrueCrypt.exe 0x91892030 4 262 1 False
2022-12-14 21:08:31 UTC N/A Disabled

The presence of **TrueCrypt** strongly suggested that **encrypted containers** might be in use, possibly hiding sensitive files such as source code.



Extracting the TrueCrypt Password from Memory

Using Volatility's TrueCrypt scanning capabilities, a password was successfully recovered from memory:

0x89ebf064 28 X2Hk2XbEJqWYsh8VdbSYg6WpG9g7

Recovered Password:

X2Hk2XbEJqWYsh8VdbSYg6WpG9g7

Why TrueCrypt Matters

TrueCrypt is a disk encryption software that allows users to create **encrypted container files** that can be mounted as virtual drives. These containers are protected by a **passphrase**, and if the system is live, that passphrase may exist in memory.

This strongly indicated that the recovered password could be used later to access protected data.

File Discovery Using Volatility CLI

Before scanning for files, the correct memory profile had to be identified.

Identifying the Profile

```
C:\Users\loki\Downloads\VolatilityWorkbench-v2.1>volatility.exe -f TrueSecrets.raw imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86_24000, Win7SP1x86
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (C:\Users\loki\Downloads\VolatilityWorkbench-v2.1\TrueSecrets.raw)
                      PAE type : PAE
                           DTB : 0x185000L
                          KDBG : 0x82732c78L
          Number of Processors : 1
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0x82733d00L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2022-12-14 21:33:30 UTC+0000
     Image local date and time : 2022-12-14 13:33:30 -0800
```

volatility.exe -f TrueSecrets.raw imageinfo

From the suggested profiles, the first one was selected:

Win7SP1x86_23418

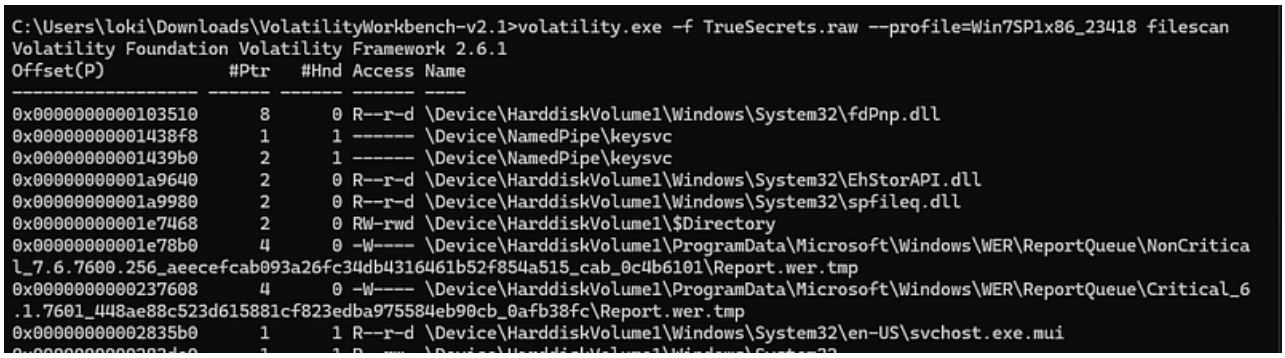
All further commands were executed using:

--profile=Win7SP1x86_23418

Scanning for Files

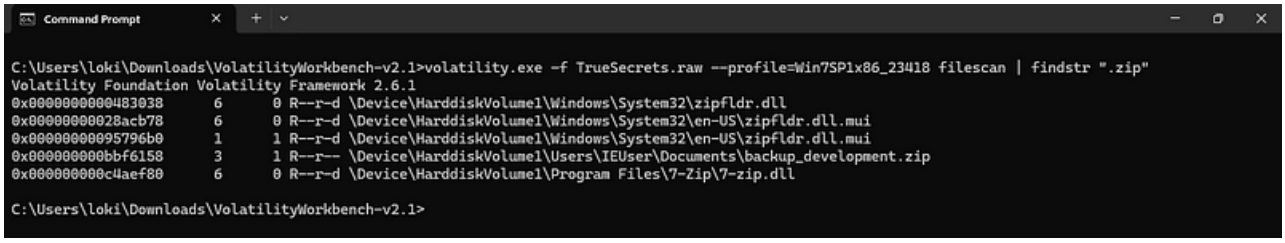
To enumerate files present in memory:

volatility.exe -f TrueSecrets.raw --profile=Win7SP1x86_23418 filescan



To narrow down results, Windows findstr was used (similar to grep in Linux):

volatility.exe -f TrueSecrets.raw --profile=Win7SP1x86_23418 filescan | findstr ".zip"



Interesting Discovery

After extensive analysis, the following file was found:

0x000000000bbf6158 3 1 R--r--
\Device\HarddiskVolume1\Users\IEUser\Documents\backup_development.zip

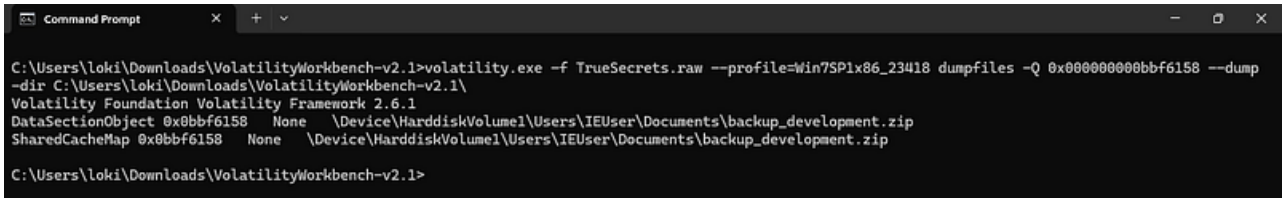
Dumping the ZIP File from Memory

To extract the file from memory, the dumpfiles plugin was used:

volatility.exe -f TrueSecrets.raw --profile=Win7SP1x86_23418 dumpfiles
-Q 0x000000000bbf6158
--dump-dir C:\Users\loki\Downloads\VolatilityWorkbench-v2.1\

Explanation:

- -Q specifies the offset of the file
- --dump-dir specifies where the extracted files will be saved

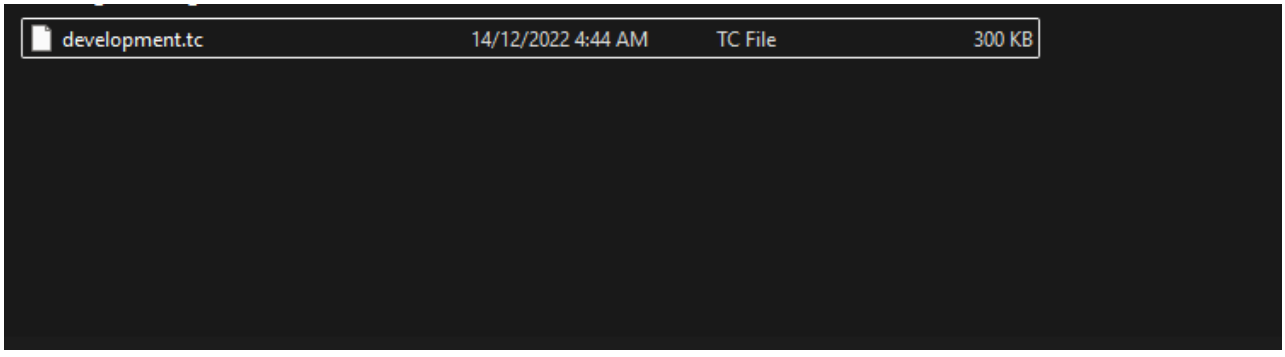


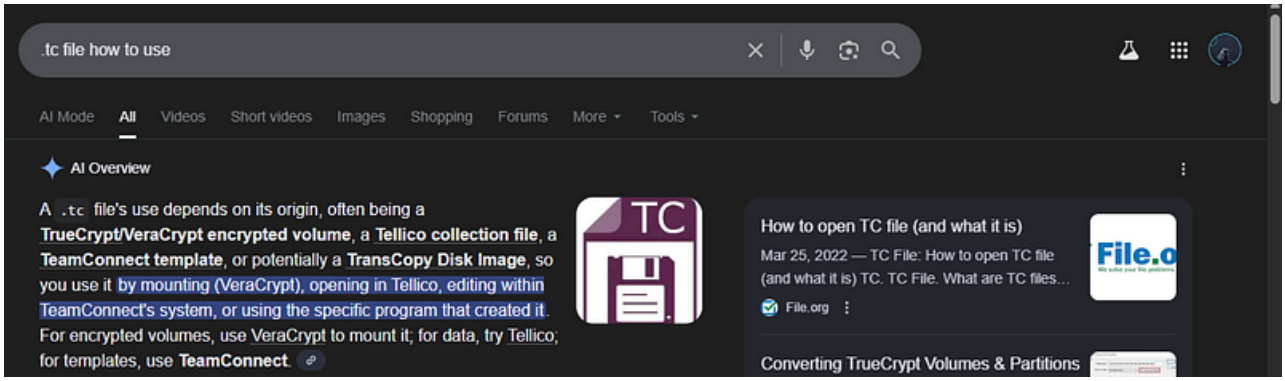
Reconstructing the ZIP Archive

The dumped output resulted in multiple .dat files. These were reconstructed into a single ZIP file:

copy /b file1.dat + file2.dat backup_reconstructed.zip

After extraction, a .tc file was discovered.



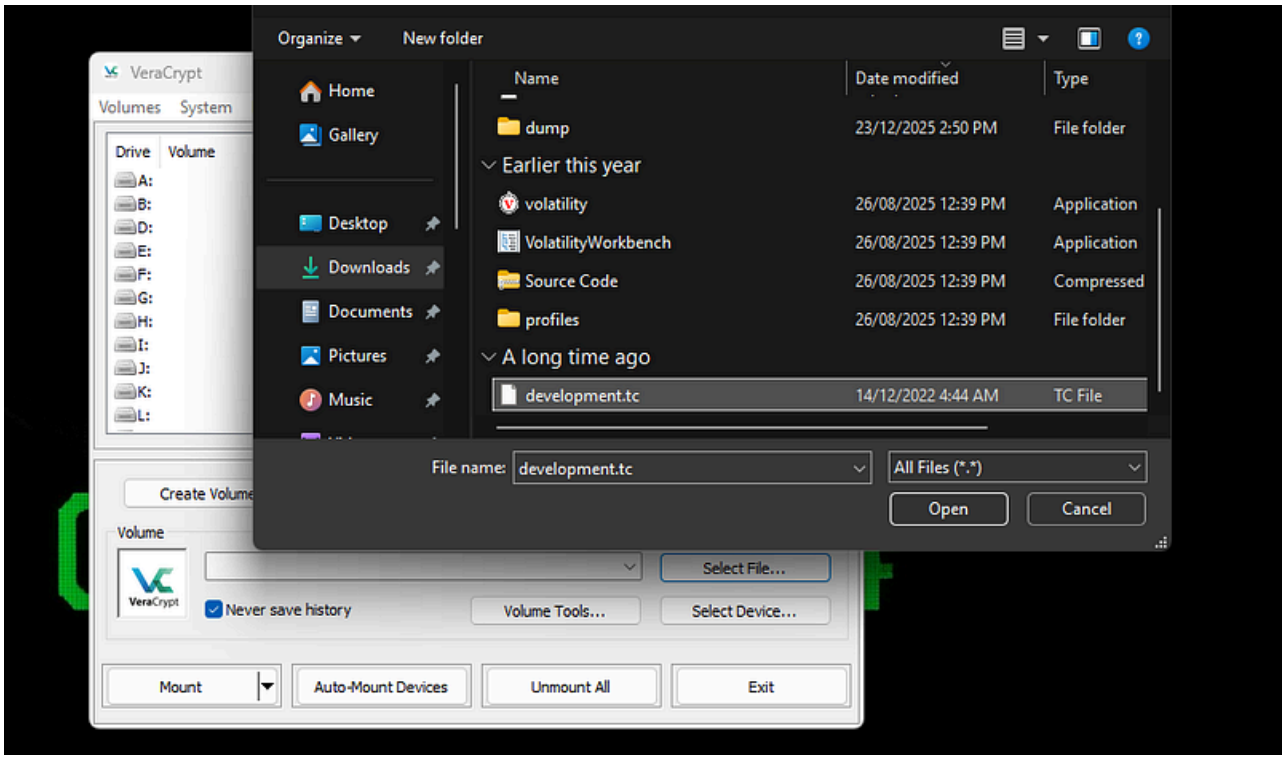


Analyzing the TrueCrypt Container

The .tc file was identified as a **TrueCrypt container**.

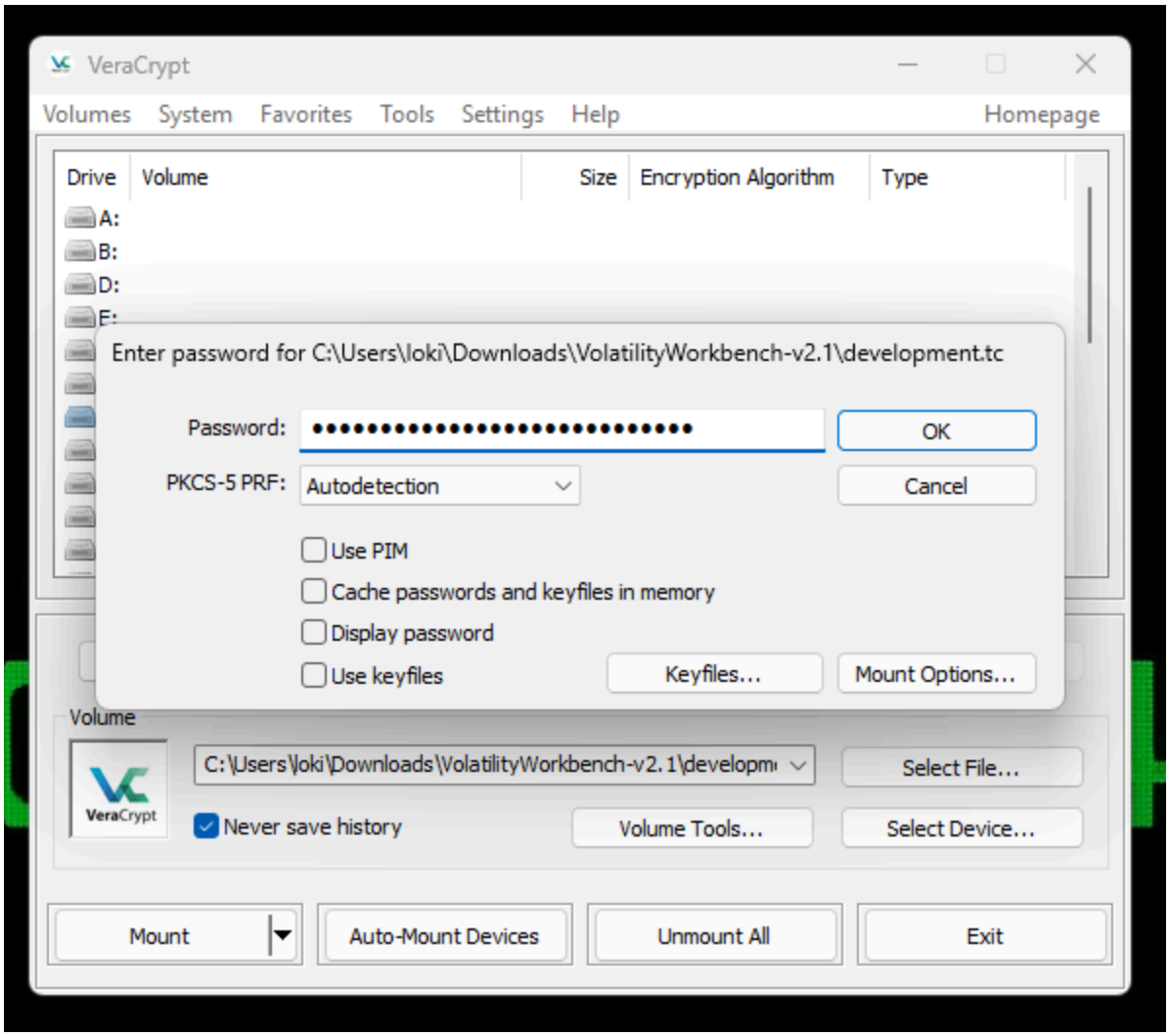
To open it, either of the following tools can be used:

- VeraCrypt
- TrueCrypt (legacy)

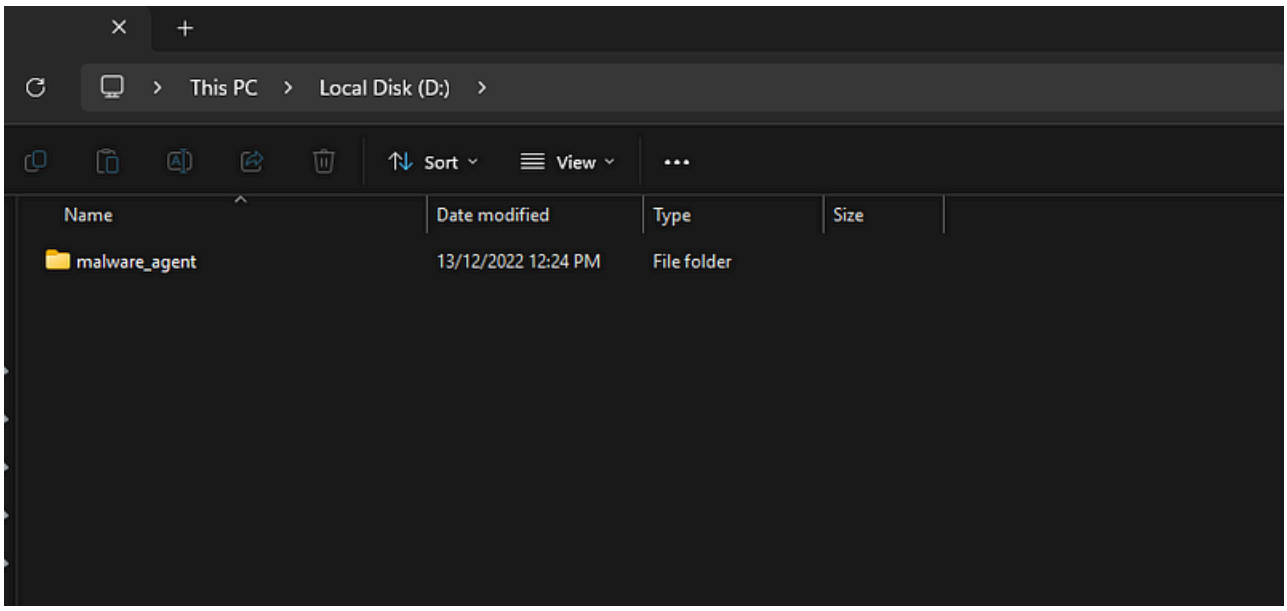


The container was mounted using:

- The recovered password:
  X2Hk2XbEJqWYsh8VdbSYg6WpG9g7

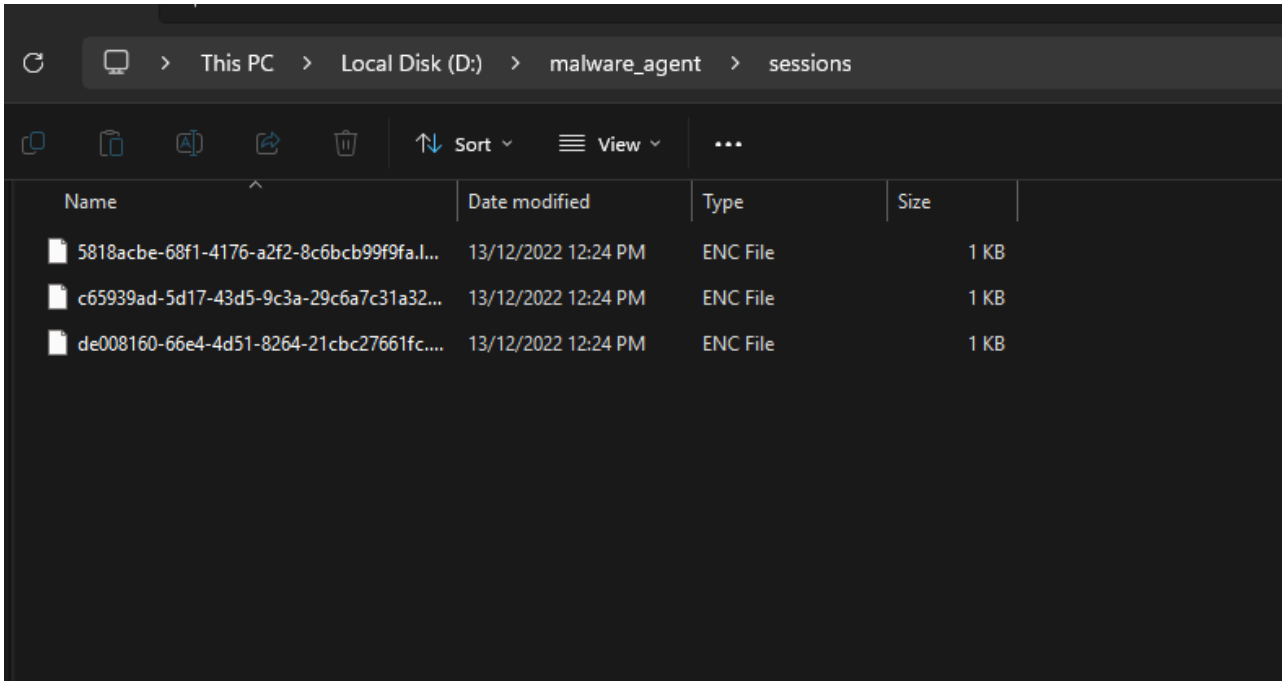


Once mounted, a file named **malware_agent** was revealed.

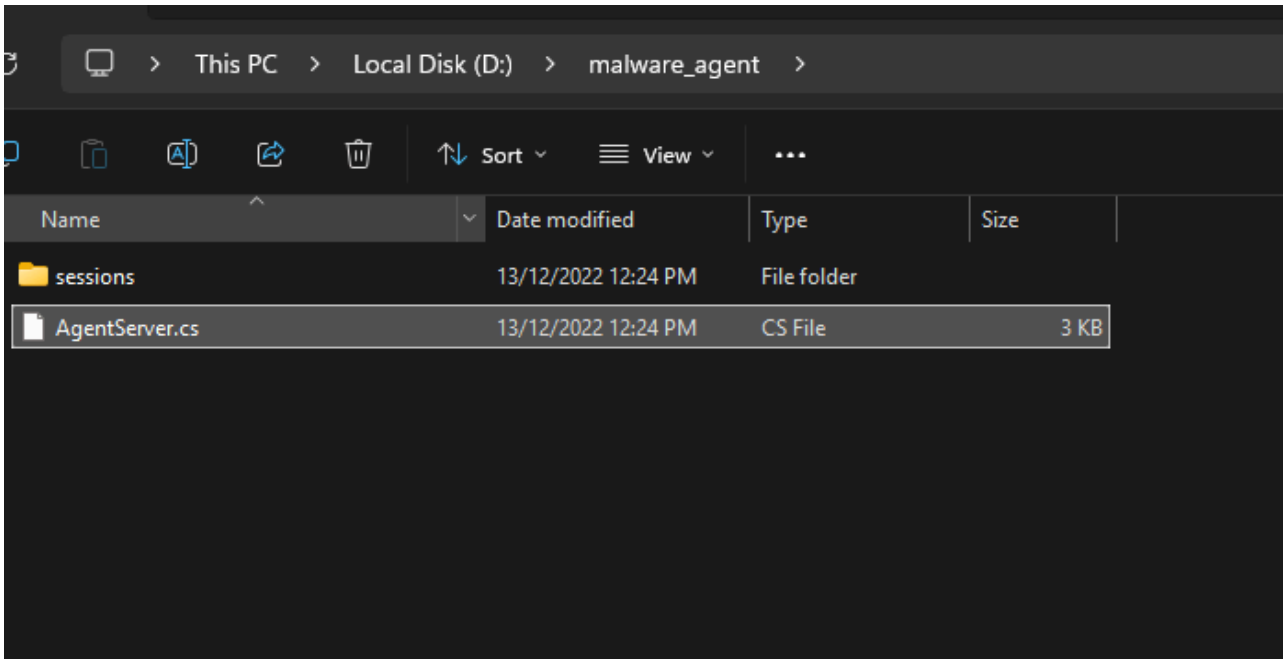


Decrypting the Malware Sessions

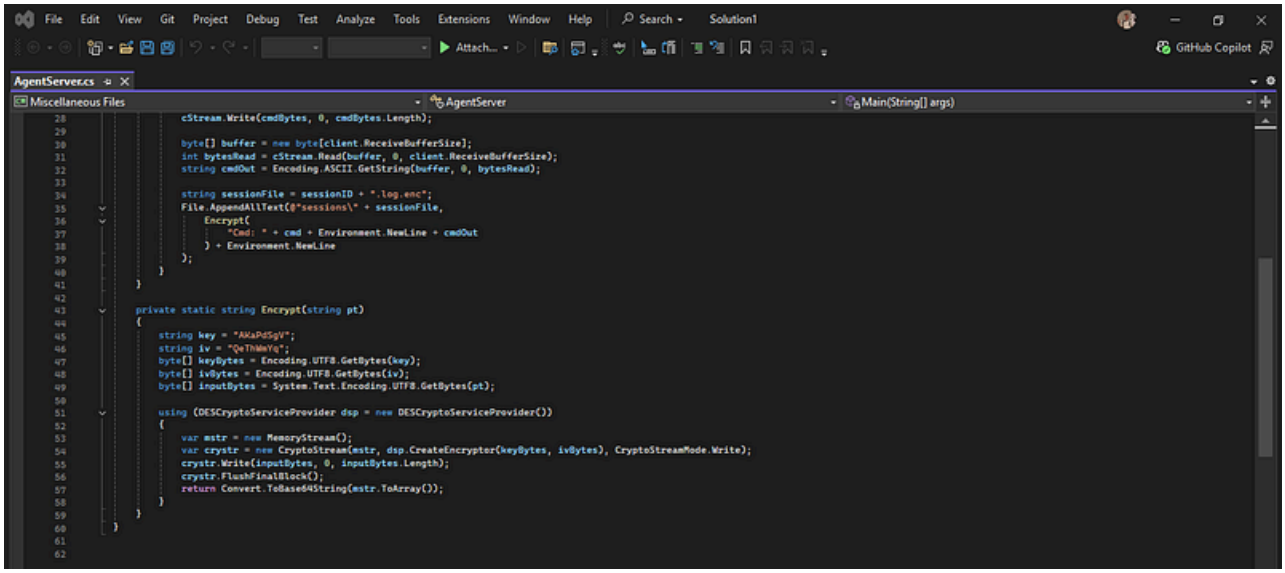
Inside the mounted drive, a sessions folder contained multiple encrypted .ENC files.

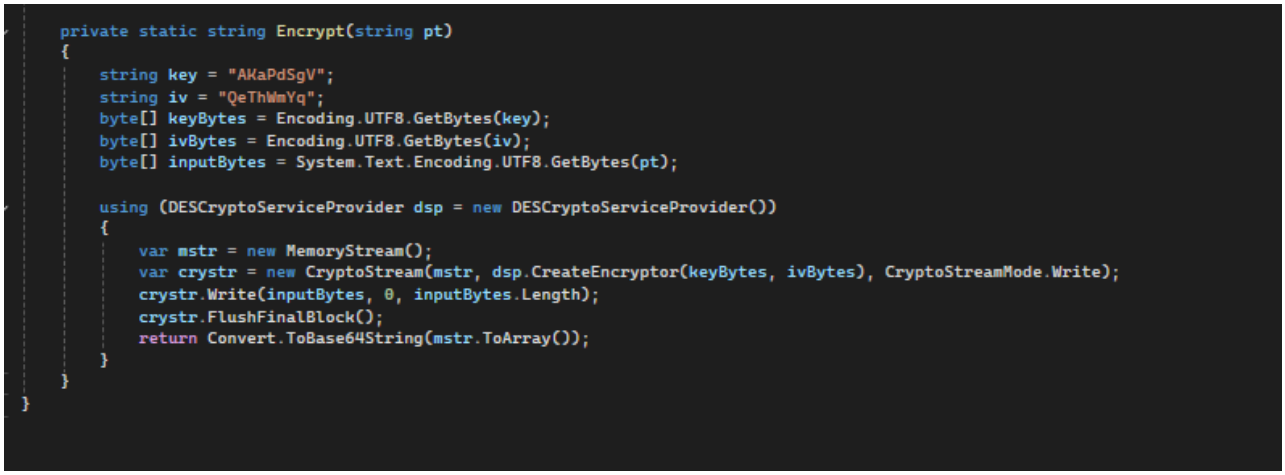


An ***AgentServer.cs*** C# file was found and opened for analysis. Inside it, the encryption parameters were revealed:




```
string key = "AKaPdSgV";
string iv  = "QeThWmYq";
```





Encryption Scheme:

Key = IV + Key

Using the key, we decrypted the encrypted session files line by line with a decryption tool.

tool:

https://anycript.com/crypto/des

Flag Recovery

After decrypting the files, the flag was found in the **third decrypted file**:

HTB{570r1ng_53cr37_1n_m3m0ry_15_n07_g00d}

References

- https://www.varonis.com/blog/how-to-use-volatility
- https://www.hackingarticles.in/memory-forensics-using-volatility-workbench/

By Alexander Sapo on December 23, 2025.

Canonical link

Exported from Medium on February 7, 2026.