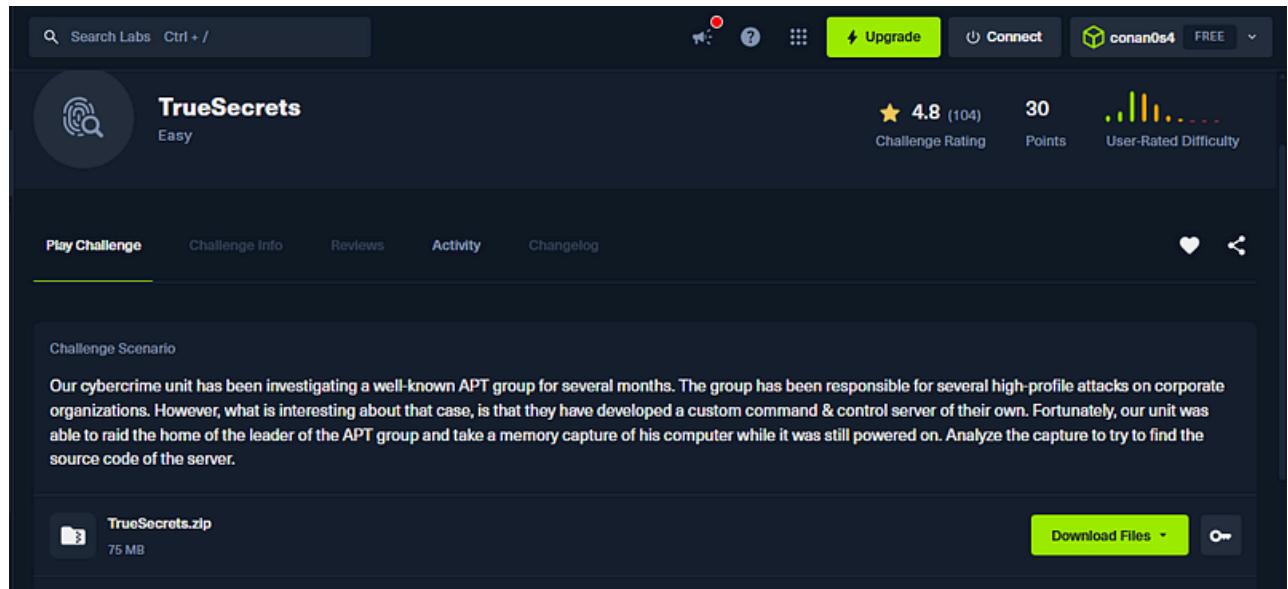


HTB Labs : TrueSecrets |Digital Forensics Write-up



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.

ID	PPID	Process Name	Image Path	Threads	Priority	Working Set	Memory Usage	Start Time	Exit Time	User	Status
864	452	svchost.exe	0x845f5030	16	399	0	False	2022-12-14 21:08:21.000000 UTC	N/A	Disabled	
904	452	svchost.exe	0x845fd28	15	311	0	False	2022-12-14 21:08:21.000000 UTC	N/A	Disabled	
928	452	svchost.exe	0x8448d28	23	956	0	False	2022-12-14 21:08:21.000000 UTC	N/A	Disabled	
992	452	svchost.exe	0x8e013488	5	114	0	False	2022-12-14 21:08:21.000000 UTC	N/A	Disabled	
1116	452	svchost.exe	0x8e0130a8	18	398	0	False	2022-12-14 21:08:21.000000 UTC	N/A	Disabled	
1228	452	spoolsv.exe	0x8e0525b0	13	275	0	False	2022-12-14 21:08:21.000000 UTC	N/A	Disabled	
1268	452	svchost.exe	0x8e4477d28	19	337	0	False	2022-12-14 21:08:21.000000 UTC	N/A	Disabled	
1352	452	taskhost.exe	0x8e0a2658	9	223	1	False	2022-12-14 21:08:22.000000 UTC	N/A	Disabled	
1448	864	dwm.exe	0x844d2d28	3	69	1	False	2022-12-14 21:08:22.000000 UTC	N/A	Disabled	
1464	1436	explorer.exe	0x8e0d3a40	32	1069	1	False	2022-12-14 21:08:22.000000 UTC	N/A	Disabled	
1636	452	svchost.exe	0x8e1023a0	10	183	0	False	2022-12-14 21:08:22.000000 UTC	N/A	Disabled	
1680	452	svchost.exe	0x8e10d998	14	224	0	False	2022-12-14 21:08:22.000000 UTC	N/A	Disabled	
1776	452	vlms.exe	0x8e07d900	4	45	0	False	2022-12-14 21:08:22.000000 UTC	N/A	Disabled	
1832	1464	VBoxTray.exe	0x83825540	12	140	1	False	2022-12-14 21:08:22.000000 UTC	N/A	Disabled	
352	452	appavc.exe	0x8e1cd8d0	4	144	0	False	2022-12-14 21:08:23.000000 UTC	N/A	Disabled	
1632	452	svchost.exe	0x8e1f6a40	5	91	0	False	2022-12-14 21:08:23.000000 UTC	N/A	Disabled	
856	452	SearchInIndexer.	0x8e0f612d0	13	626	0	False	2022-12-14 21:08:28.000000 UTC	N/A	Disabled	
2128	1464	TrueCrypt.exe	0x91892030	4	262	1	False	2022-12-14 21:08:31.000000 UTC	N/A	Disabled	
2760	452	svchost.exe	0x91865790	13	362	0	False	2022-12-14 21:10:23.000000 UTC	N/A	Disabled	
2332	584	WmiPrvSE.exe	0x83911848	5	112	0	False	2022-12-14 21:12:23.000000 UTC	N/A	Disabled	
2580	452	taskhost.exe	0x8e1ef208	5	86	1	False	2022-12-14 21:13:01.000000 UTC	N/A	Disabled	
2176	1464	7zFM.exe	0x8382f198	3	135	1	False	2022-12-14 21:22:44.000000 UTC	N/A	Disabled	
3212	1464	DumpIt.exe	0x83c1d030	2	38	1	False	2022-12-14 21:33:28.000000 UTC	N/A	Disabled	
272	368	conhost.exe	0x83c0a030	2	34	1	False	2022-12-14 21:33:28.000000 UTC	N/A	Disabled	

Time Stamp: Tue Dec 23 12:59:14 2025

Activate Windows
Go to Settings to activate Windows

Clear Log Save to file Copy to clipboard About Exit

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 : 0xfffff0000L
          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

```
C:\Users\loki\Downloads\VolatilityWorkbench-v2.1>volatility.exe -f TrueSecrets.raw --profile=Win7SP1x86_23418 filescan
Volatility Foundation Volatility Framework 2.6.1
Offset(P)      #Ptr  #Hnd Access Name
-----
0x00000000000103510  8     0 R--r-d \Device\HarddiskVolume1\Windows\System32\fdPnp.dll
0x000000000001438f8  1     1 ----- \Device\NamedPipe\keysvc
0x000000000001439b0  2     1 ----- \Device\NamedPipe\keysvc
0x000000000001a9640  2     0 R--r-d \Device\HarddiskVolume1\Windows\System32\EhStorAPI.dll
0x000000000001a9980  2     0 R--r-d \Device\HarddiskVolume1\Windows\System32\spfileq.dll
0x000000000001e7468  2     0 RW-rwd \Device\HarddiskVolume1\$Directory
0x000000000001e78b0  4     0 -W---- \Device\HarddiskVolume1\ProgramData\Microsoft\Windows\WER\ReportQueue\NonCritica
l_6.7600.256_aeecefca893a26fc34db4316461b52f854515_cab_0c4b6101\Report.wer.tmp
0x00000000000237608  4     0 -W---- \Device\HarddiskVolume1\ProgramData\Microsoft\Windows\WER\ReportQueue\Critical_6
.1.7601_448ae88c523d615881cf823edba975584eb90cb_0afbf38fc\Report.wer.tmp
0x000000000002835b0  1     1 R--r-d \Device\HarddiskVolume1\Windows\System32\en-US\svchost.exe.mui
0x00000000000283740  1     1 R--r-d \Device\HarddiskVolume1\Windows\System32\en-US\svchost.exe.mui
```

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

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

```
C:\Users\loki\Downloads\VolatilityWorkbench-v2.1>volatility.exe -f TrueSecrets.raw --profile=Win7SP1x86_23418 filescan | findstr ".zip"
Volatility Foundation Volatility Framework 2.6.1
0x00000000000483038  6     0 R--r-d \Device\HarddiskVolume1\Windows\System32\zipfldr.dll
0x000000000028acb78  6     0 R--r-d \Device\HarddiskVolume1\Windows\System32\en-US\zipfldr.dll.mui
0x000000000095796b0  1     1 R--r-d \Device\HarddiskVolume1\Windows\System32\en-US\zipfldr.dll.mui
0x0000000000bf6158   3     1 R--r-- \Device\HarddiskVolume1\Users\IEUser\Documents\backup_development.zip
0x00000000004aeff80  6     0 R--r-d \Device\HarddiskVolume1\Program Files\7-Zip\7-zip.dll

C:\Users\loki\Downloads\VolatilityWorkbench-v2.1>
```

Interesting Discovery

After extensive analysis, the following file was found:

0xooooooooooooobbf6158 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

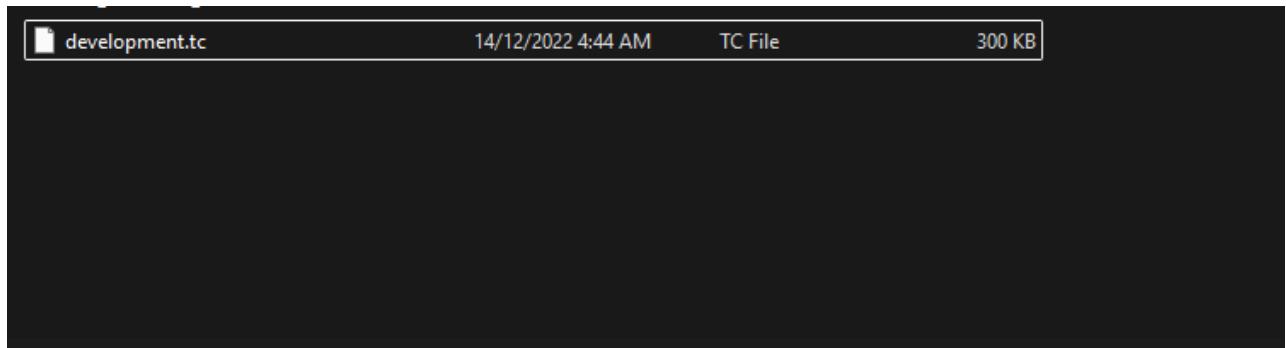
```
C:\Users\loki\Downloads\VolatilityWorkbench-v2.1>volatility.exe -f TrueSecrets.raw --profile=Win7SP1x86_23418 dumpfiles -Q 0x000000000bbf6158 --dump
-dir C:\Users\loki\Downloads\VolatilityWorkbench-v2.1\
Volatility Foundation Volatility Framework 2.6.1
DataSectionObject 0x0bbf6158 None \Device\HarddiskVolume1\Users\IEUser\Documents\backup_development.zip
SharedCacheMap 0x0bbf6158 None \Device\HarddiskVolume1\Users\IEUser\Documents\backup_development.zip
C:\Users\loki\Downloads\VolatilityWorkbench-v2.1>
```

Reconstructing the ZIP Archive

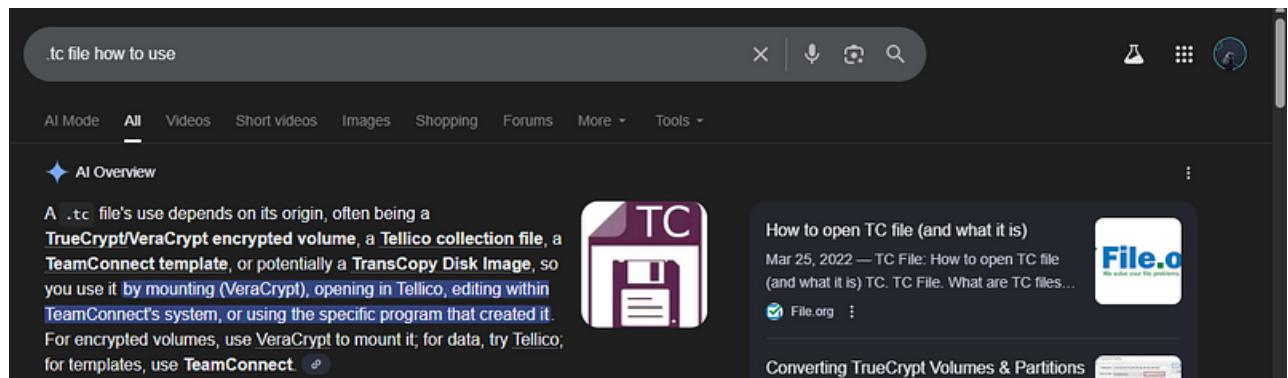
The dumped output resulted in multiple .dat files. These were reconstructed into a single ZIP file:

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

After extraction, a .tc file was discovered.



backup_development	23/12/2025 2:18 PM	Compressed (zipp...)	816 KB
file.None.0x9185db40.vacb	23/12/2025 2:12 PM	VACB File	512 KB
file.None.0x839339d0.dat	23/12/2025 2:12 PM	DAT File	304 KB

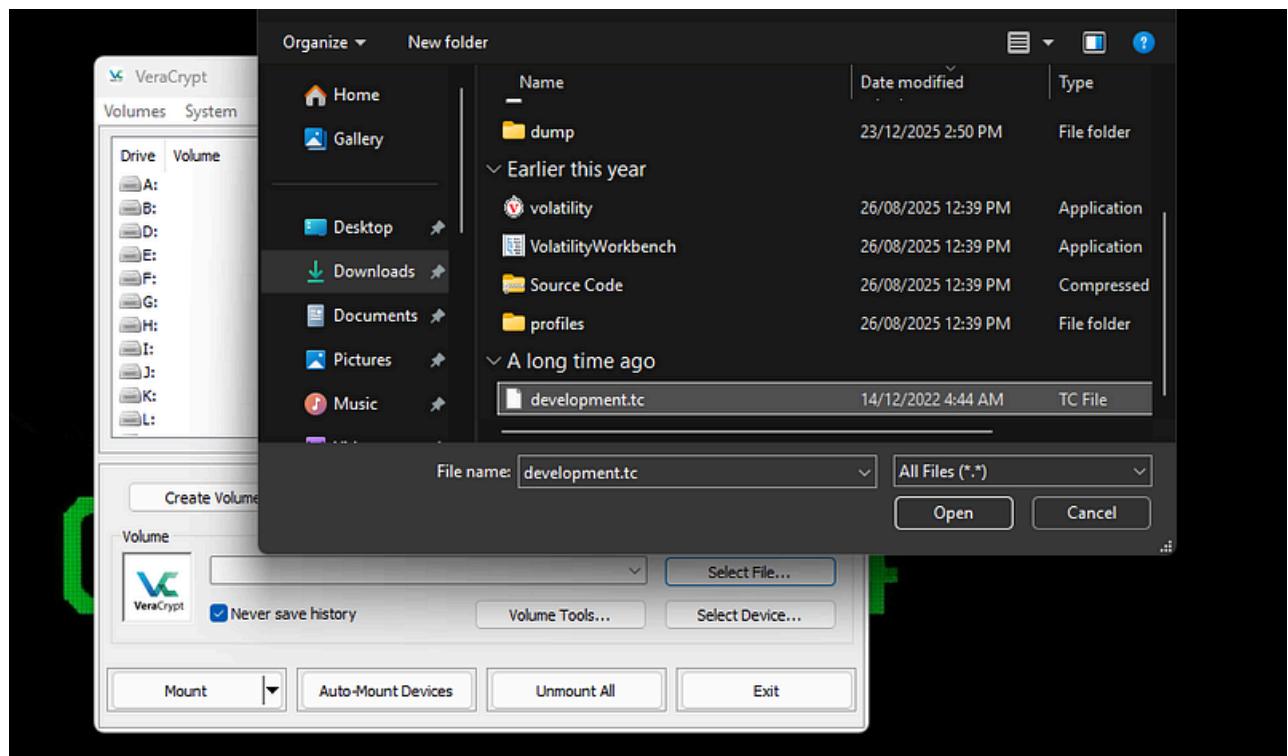


Analyzing the TrueCrypt Container

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

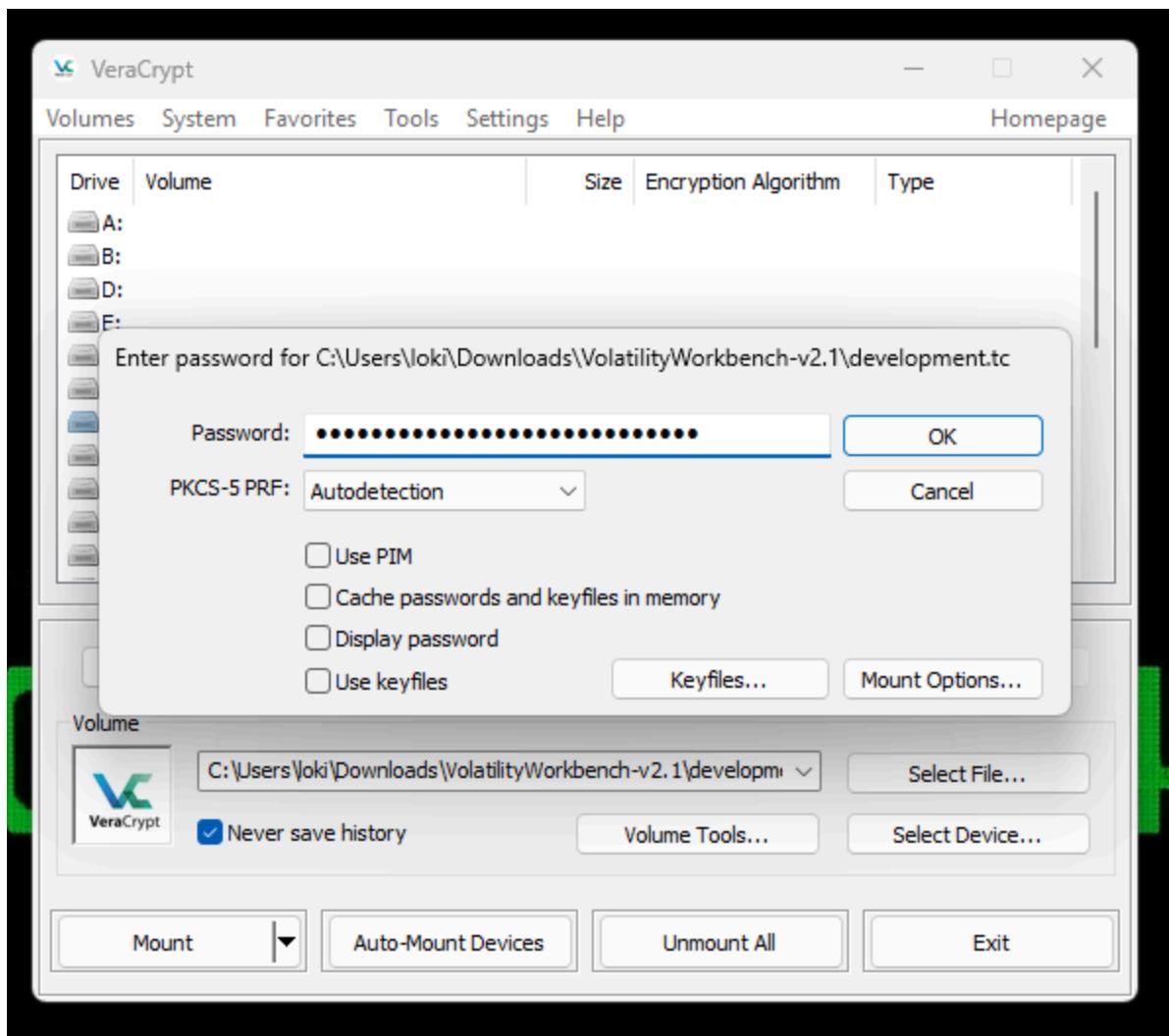
To open it, either of the following tools can be used:

- VeraCrypt
- TrueCrypt (legacy)

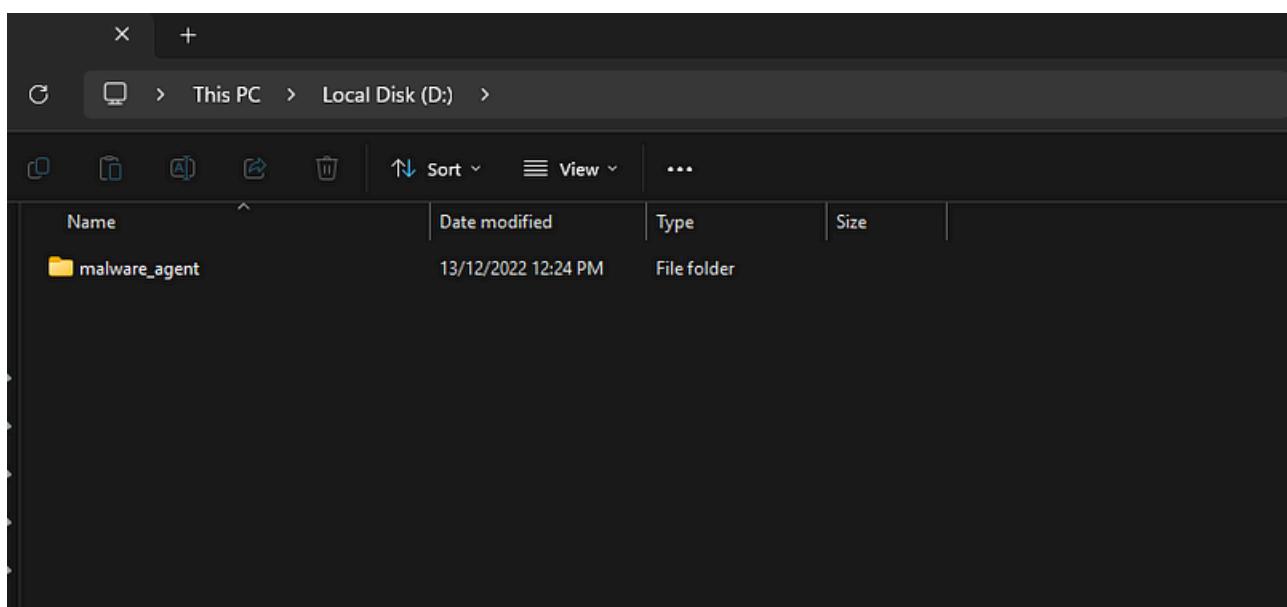


The container was mounted using:

- The recovered password:
X2Hk2XbEJqWYsh8VdbSYg6WpG9g7

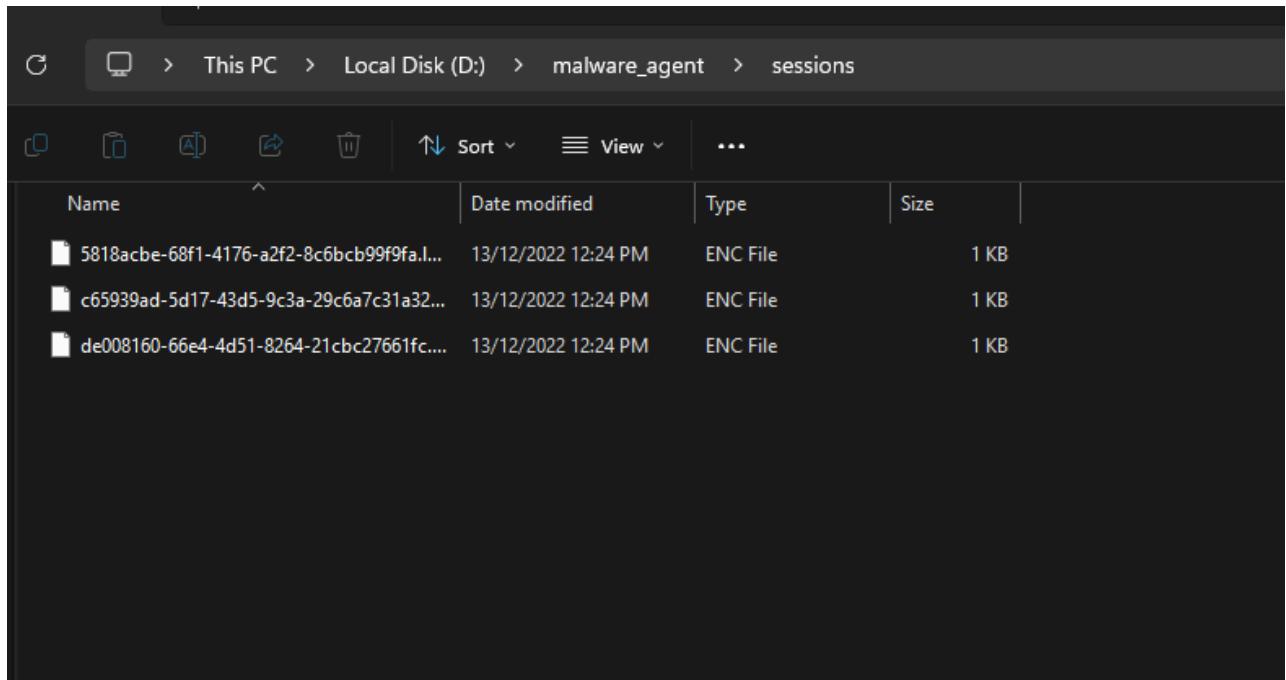


Once mounted, a file named **malware_agent** was revealed.

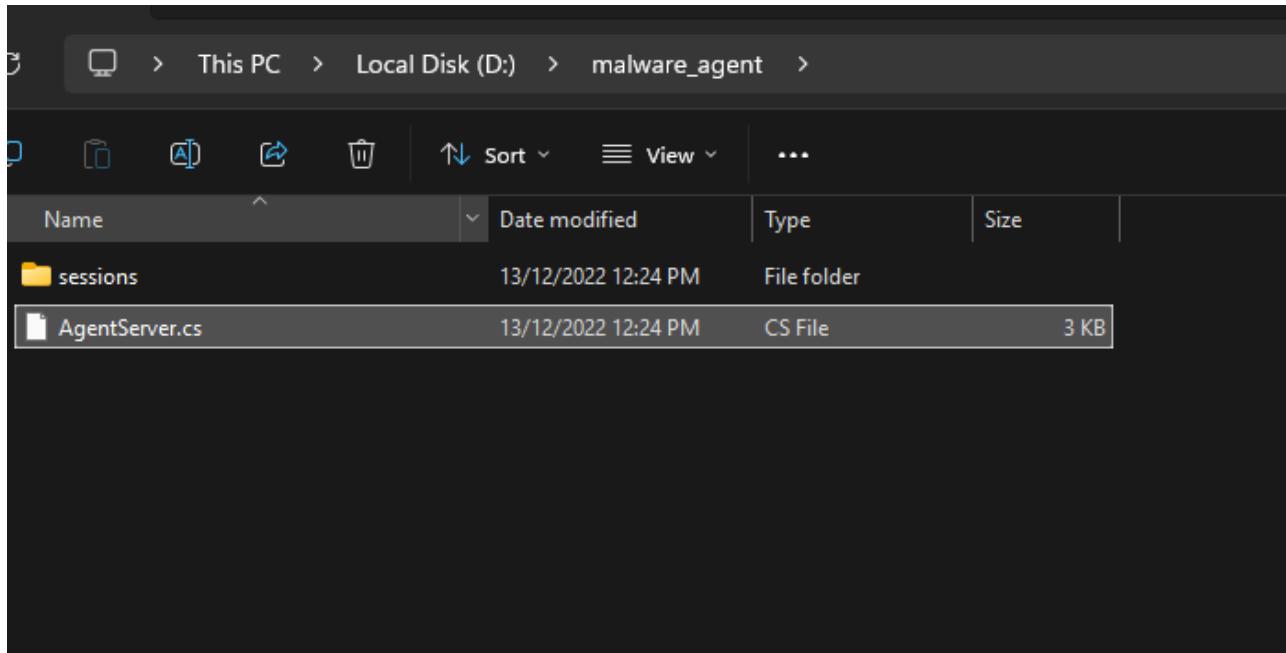


Decrypting the Malware Sessions

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";
```

```

File Edit View Git Project Debug Test Analyze Tools Extensions Window Help Search Solution1 GitHub Copilot
AgentServer.cs -> X
Miscellaneous Files -> AgentServer -> Main(String[] args)
28     cStream.Write(cmdBytes, 0, cmdBytes.Length);
29
30     byte[] buffer = new byte[client.ReceiveBufferSize];
31     int bytesRead = cStream.Read(buffer, 0, client.ReceiveBufferSize);
32     string cmdOut = Encoding.ASCII.GetString(buffer, 0, bytesRead);
33
34     string sessionId = sessionId + ".log.anc";
35     File.AppendAllText(@"sessions\" + sessionId,
36         Encrypt(
37             "Cmd:" + cmd + Environment.NewLine + cmdOut
38         ) + Environment.NewLine
39     );
40 }
41
42 private static string Encrypt(string pt)
43 {
44     string key = "AKaPdSgV";
45     string iv = "QeThWmYq";
46     byte[] keyBytes = Encoding.UTF8.GetBytes(key);
47     byte[] ivBytes = Encoding.UTF8.GetBytes(iv);
48     byte[] inputBytes = System.Text.Encoding.UTF8.GetBytes(pt);
49
50     using (DESCryptoServiceProvider dsp = new DESCryptoServiceProvider())
51     {
52         var mstr = new MemoryStream();
53         var crystr = new CryptoStream(mstr, dsp.CreateEncryptor(keyBytes, ivBytes), CryptoStreamMode.Write);
54         crystr.Write(inputBytes, 0, inputBytes.Length);
55         crystr.FlushFinalBlock();
56         return Convert.ToBase64String(mstr.ToArray());
57     }
58 }
59
60 }
61
62

```

```

private static string Encrypt(string pt)
{
    string key = "AKaPdSgV";
    string iv = "QeThWmYq";
    byte[] keyBytes = Encoding.UTF8.GetBytes(key);
    byte[] ivBytes = Encoding.UTF8.GetBytes(iv);
    byte[] inputBytes = System.Text.Encoding.UTF8.GetBytes(pt);

    using (DESCryptoServiceProvider dsp = new DESCryptoServiceProvider())
    {
        var mstr = new MemoryStream();
        var crystr = new CryptoStream(mstr, dsp.CreateEncryptor(keyBytes, ivBytes), CryptoStreamMode.Write);
        crystr.Write(inputBytes, 0, inputBytes.Length);
        crystr.FlushFinalBlock();
        return Convert.ToBase64String(mstr.ToArray());
    }
}

```

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_m3mory_15_no7_good}

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.