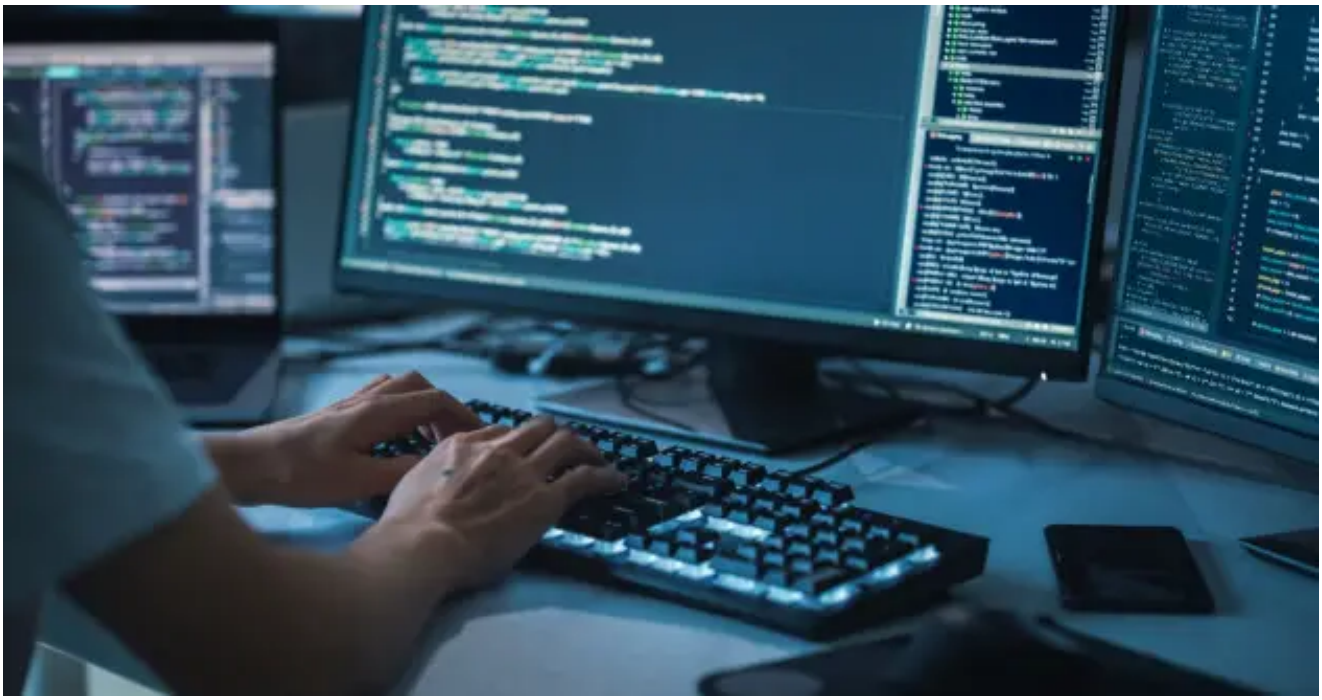
When the Absence of Noise Becomes Signal: Defensive Considerations for Lazarus FudModule

securityintelligence.com/posts/defensive-considerations-lazarus-fudmodule/

Home / Endpoint



Endpoint March 20, 2023

By John Dwyer 7 min read

In February 2023, X-Force posted a blog entitled "Direct Kernel Object Manipulation (DKOM) Attacks on ETW Providers" that details the capabilities of a sample attributed to the Lazarus group leveraged to impair visibility of the malware's operations. This blog will not rehash analysis of the Lazarus malware sample or Event Tracing for Windows (ETW) as that has been previously covered in the X-Force blog post. This blog will focus on highlighting the opportunities for detection of the FudModule within the Lazarus sample analyzed by X-Force, as well as highlight a strategy of using telemetry stream health as a mode of detecting malware designed to impair defenses through ETW tampering.

One Ring 0 To Rule Them All

The Lazarus FudModule begins with the installation of a Dell driver that is vulnerable to CVE-2021-21551 which allows the malware to elevate privileges to a level where DKOM attacks are possible. This type of attack is referred to as a bring your own vulnerable driver (BYOVD) attack. In a BYOVD attack, an attacker installs a driver that is vulnerable to an exploit that enables the attacker to cross the boundary from administrative access to ring 0 or kernel-

mode access. Ring 0 access enables the attacker to bypass or disable security technology and evade detection by security professionals by operating deeper within the operating system.

Can’t Hit What You Can’t See

As detailed in the X-Force blog, after obtaining kernel mode privileges the FudModule begins targeting kernel structures to impair telemetry sources on the host by targeting Event Tracing for Windows (ETW) registration handles. ETW registration handles are used to retrieve configuration information for a specific provider, the handle can test whether a provider is enabled for specific keywords or information levels. Additionally, ETW registration handles are used to call event tracing and logging functions for a specific provider. The FudModule leverages the nt!EtwRegister function to enumerate entries associated with the RegHandle parameter and then updates the value with NULL effectively disabling all system ETW providers for all consuming applications, including those providers used by some antivirus and endpoint detection and response (EDR) solutions.

Dynamic Detection Opportunity Through Monitoring ETW Data Stream Health

While the FudModule’s capability to tamper with ETW registration handles does enable it to create and operate within a blind spot for security tooling, this is not the first piece of malware X-Force has encountered that tampers with data streams to avoid detection. To handle these types of detection challenges, X-Force has developed a detection method focused on identification of malicious activity by testing data stream health.

The following section will demonstrate a proof-of-concept ETW data stream monitor like X-Force’s using PowerShell and Symon so organizations can start considering a similar detection capability.

The FudModule’s attack against ETW registration handles may disable ETW monitoring providers, but X-Force was not able to uncover evidence that the malware prevents new ETW registration handles from being created and used. Defenders can leverage this opportunity to register a new ETW session to an existing security-related ETW provider, create test data, and compare the telemetry between the new ETW session and the existing ETW session. Organizations can implement a similar strategy to detect when there is a discrepancy between the telemetry registered in the data source and raise an alarm regarding security-related ETW data stream health when a discrepancy is observed.

When Microsoft deployed PatchGuard in the early 2000s, it limited the ability for security vendors to hook the kernel directly to detect malware executing on a system. Since the introduction of PatchGuard, vendors have leveraged Kernel callback functions to monitor

activities on a system. For a comprehensive list of Kernel callback functions here, see the Kernel CallbackFunctions list created by CodeMachine here.

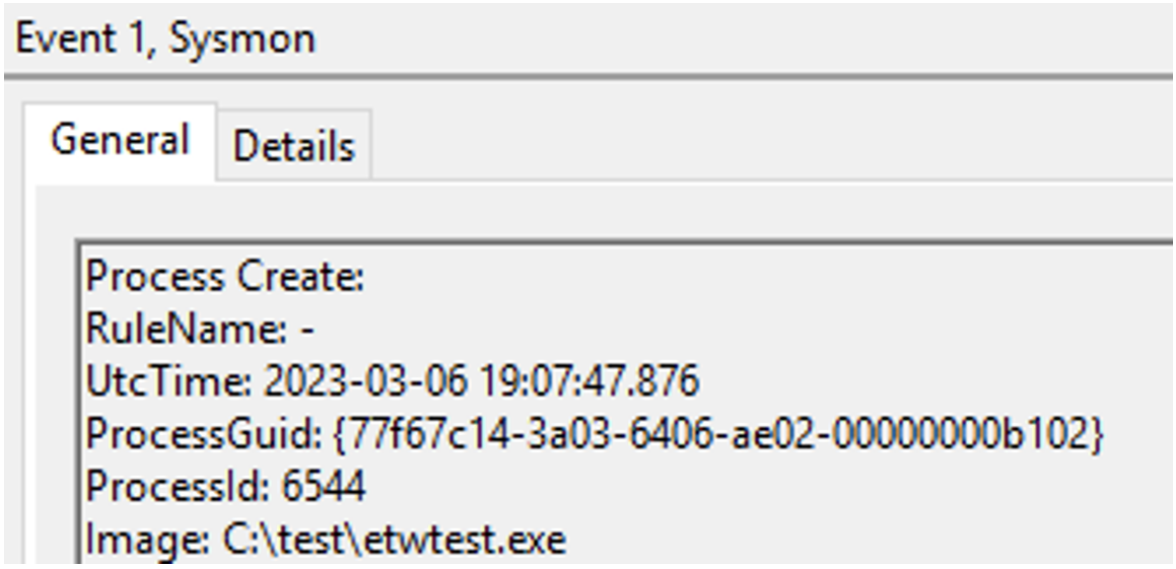
The most common Kernel callback functions leveraged by security vendors are:

- PsSetCreateProcessNotifyRoutine
- CmRegisterCallbackEx
- PsSetCreateThreadNotifyRoutine
- ObRegisterCallbacks
- PsSetLoadImageNotifyRoutine

For the purposes of this blog, we will be focusing on PsSetCreateProcessNotifyRoutine which is a callback routine to monitor process creation activities performed by APIs like CreateProcess(), CreateProcessAsUser(), CreateProcessWithToken(), and CreateProcessWithLogon().

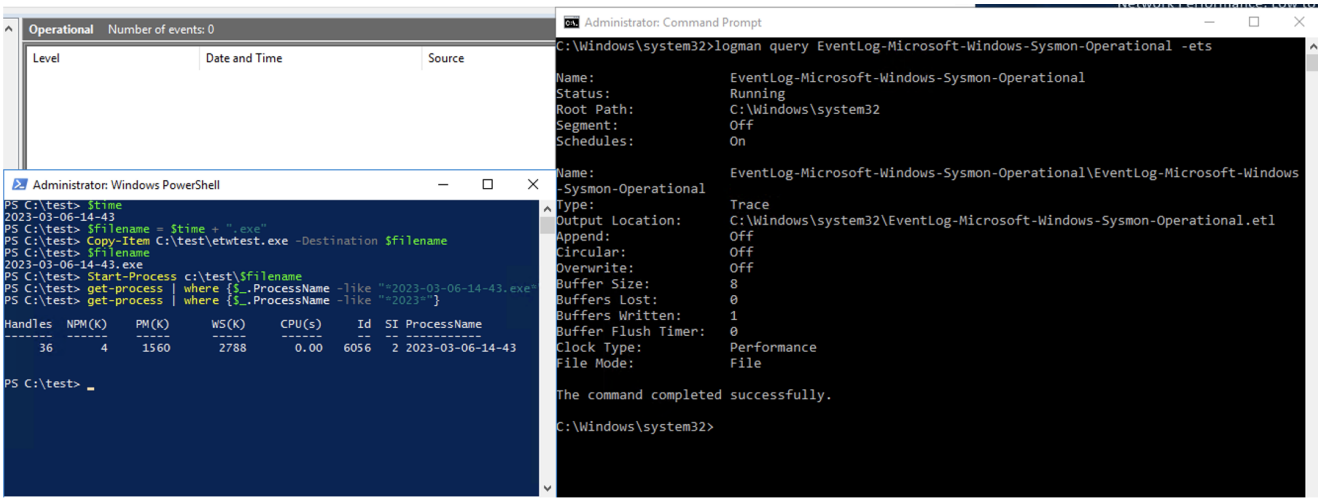
Security products often leverage PsSetCreateProcessNotifyRoutine as a mechanism to be notified when a user spawns a new process via the CreateProcessW() function in kernel32.dll. With the introduction of kernel-mode drivers for modern security products, they are now able to intercept and inject into resulting process creation events by hooking ntdll.dll to perform inspections of the process activity.

To simulate the resulting blind spots from the FudModule, X-Force will disrupt the data streams enabling Sysmon to monitor process creation events via PsSetCreateProcessNotifyRoutine without stopping the Sysmon ETW Trace Session or removing the ETW Provider.



Sysmon Operating Normally

To simulate the BYOVD attack, X-Force leveraged PsExec to assume privileges of NT Authority\SYSTEM and modified the Sysmon ETW Trace configuration so that while the System ETW Trace Session is running, no data will be written to the Event Log.



Sysmon After Trace Session Tampering

At this point, the Sysmon service is running, the ETW Trace is running, and the Sysmon ETW provider is available, but no process creation events are being written to the Event Log effectively simulating the behavior of the FudModule for a single ETW data source on an infected system.

As previously stated, the FudModule does not appear to remove providers or continually null value ETW registration handles, so Defenders now have an opportunity to test ETW data stream health and be alerted when security telemetry is not operating as expected either through corruption or malicious activity. Fortunately, Microsoft makes this process easy with a default ETW provider called Microsoft-Windows-Kernel-Process which handles the execution of all threads in a process and directly interacts with the PsSetCreateProcessNotifyRoutine routine.

Within the Microsoft-Windows-Kernel-Process provider, various data sources are available including new process creation, process thread, image loads, and CPU priority. Using the logman utility, defenders can easily create a new ETW Trace Session to collect new process creation events.

```
C:\Windows\system32>logman query providers Microsoft-Windows-Kernel-Process

Provider                                GUID
-------------------------------------------------------------------------------
Microsoft-Windows-Kernel-Process        {22FB2CD6-0E7B-422B-A0C7-2FAD1FD0E716}

Value               Keyword             Description
-------------------------------------------------------------------------------
0x0000000000000010  WINEVENT_KEYWORD_PROCESS
0x0000000000000020  WINEVENT_KEYWORD_THREAD
0x0000000000000040  WINEVENT_KEYWORD_IMAGE
0x0000000000000080  WINEVENT_KEYWORD_CPU_PRIORITY
0x0000000000000100  WINEVENT_KEYWORD_OTHER_PRIORITY
0x0000000000000200  WINEVENT_KEYWORD_PROCESS_FREEZE
0x0000000000000400  WINEVENT_KEYWORD_JOB
0x0000000000000800  WINEVENT_KEYWORD_ENABLE_PROCESS_TRACING_CALLBACKS
0x0000000000001000  WINEVENT_KEYWORD_JOB_IO
0x0000000000002000  WINEVENT_KEYWORD_WORK_ON_BEHALF
0x8000000000000000  Microsoft-Windows-Kernel-Process/Analytic

Value               Level               Description
-------------------------------------------------------------------------------
0x04                win:Informational   Information
```

```
C:\Windows\system32>logman create trace xftrace -ets
The command completed successfully.

C:\Windows\system32>logman update xftrace -p Microsoft-Windows-Kernel-Process 0x10 -ets
The command completed successfully.

C:\Windows\system32>logman query xftrace -ets

Name:                 xftrace
Status:               Running
Root Path:            C:\Windows\system32
Segment:              Off
Schedules:            On

Name:                 xftrace\xftrace
Type:                 Trace
Output Location:      C:\Windows\system32\xftrace.etl
Append:               Off
Circular:             Off
Overwrite:            Off
Buffer Size:          8
Buffers Lost:         0
Buffers Written:      1
Buffer Flush Timer:   0
Clock Type:           Performance
File Mode:            File

Provider:
Name:                 Microsoft-Windows-Kernel-Process
Provider Guid:        {22FB2CD6-0E7B-422B-A0C7-2FAD1FD0E716}
Level:                255
KeywordsAll:          0x0
KeywordsAny:          0x10 (WINEVENT_KEYWORD_PROCESS)
Properties:           64
Filter Type:          0

The command completed successfully.
```

Creating a New ETW Trace for Process Creation Events

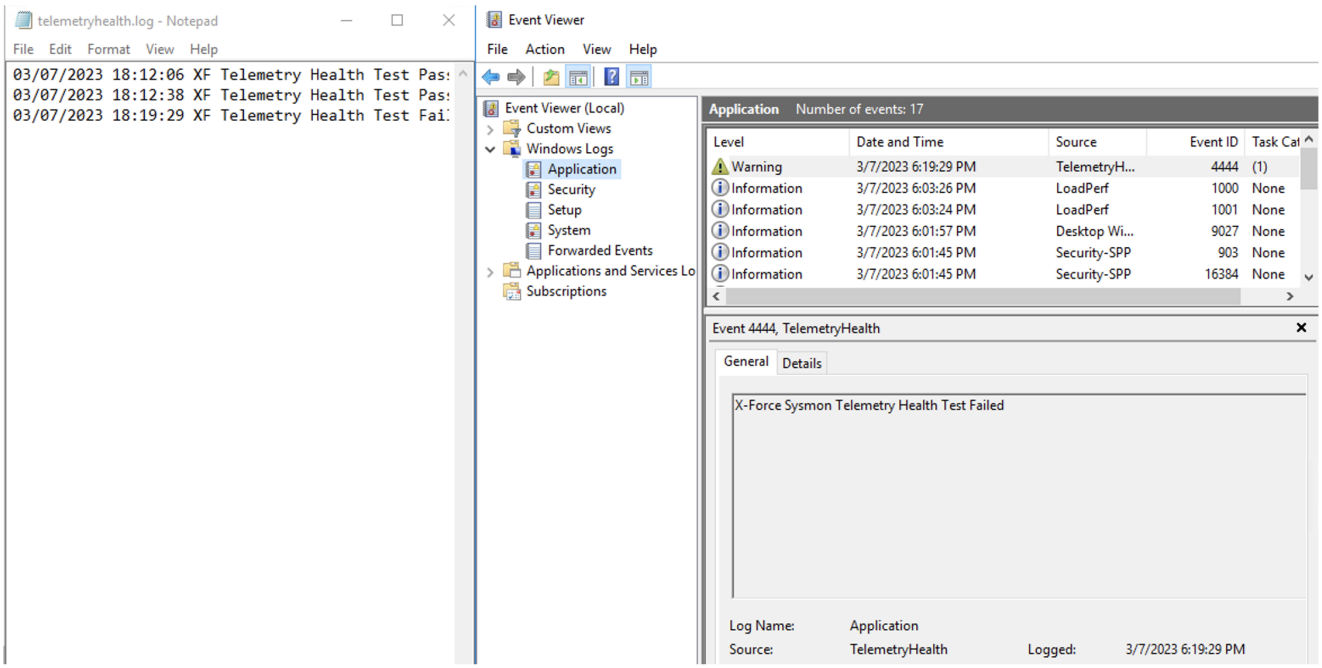
The following PowerShell is a proof-of-concept replicating the core capabilities of X-Force's telemetry stream health tests for threat detection. In this script, a new ETW Trace session is created and subscribed to Microsoft-Windows-Kernel-Process WINEVENT_KEYWORD_PROCESS events. The script then creates test data using the current timestamp as a unique identifier and checks whether the test data was located within the newly created ETW Trace session and Sysmon. When the test data is not found in the Sysmon log, an alert is written to the Application Event Log which can be alerted upon for defenders.

ETWHealthCheack.ps1

```powershell
#Create Unique Bin for Testing
$time = (Get-Date).ToString("yyyy-MM-dd-mm-ss");$filename = $time + ".exe";Copy-Item C:\test\etwtest.exe -Destination c:\test\$filename
#Create New ETW Trace Session for Kernel Process
logman create trace xftrace -ets -ow -bs 1;logman update xftrace -p Microsoft-Windows-Kernel-Process 0x10 -ets;Start-Sleep -Seconds 2
#Create Test Telemetry
start-process c:\test\$filename;Start-Sleep -Seconds 2;$killname = $time.ToString();stop-process -Name $killname;Start-Sleep -Seconds 2;remove-item c:\test\$filename
#Search Sysmon and ETL for Test Telemetry
if(Get-WinEvent -logname Microsoft-Windows-Sysmon/Operational | where {($_.TimeCreated -gt (Get-Date).AddMinutes(-1)) -and $_.Message -like "*$time*" -and $_.Id -eq 1  })
{
    "$([datetime]::Now) XF Telemetry Health Test Passed" | Out-File -FilePath "c:\test\telemetryhealth.log" -Append
}
else
{
    $runloop = $true
    while ($runloop -eq $true)
    {
        $FileSize = (Get-Item -Path .\xftrace.etl).Length
        if ($FileSize/1kb -gt 1)
        {
          if(Get-WinEvent -Path .\xftrace.etl -oldest | where {$_.Message -like "*$time*"})
                {
                    if([System.Diagnostics.EventLog]::SourceExists("TelemetryHealth"))
                    {

                        "$([datetime]::Now) XF Telemetry Health Test Failed" | Out-File -FilePath "c:\test\telemetryhealth.log" -Append
                        Write-EventLog -LogName "Application" -Source "TelemetryHealth" -EventID 4444 -EntryType Warning -Message "X-Force Sysmon Telemetry Health Test Failed"
                        $runloop = $false
                    }
                    else
                    {
                         "$([datetime]::Now) XF Telemetry Health Test Failed" | Out-File -FilePath "c:\test\telemetryhealth.log" -Append
                         New-EventLog -source  TelemetryHealth -LogName Application
                         Write-EventLog -LogName "Application" -Source "TelemetryHealth" -EventID 4444 -EntryType Warning -Message "X-Force Sysmon Telemetry Health Test Failed"
                         $runloop = $false
                    }
                }
        }
        else
        {
            write-host "ETL not updated, looping in 5 seconds"
            start-sleep -Seconds 5
        }
    }
}
#clean up ETW trace
logman stop xftrace -ets
```

ETW Data Health Test Script



Create Alert When Telemetry Health Test Fails

The following sections will detail static detection opportunities for the Lazarus FudModule based on the analysis of the malware sample.

Static Detection Opportunities Within File Activity

To initiate the BYOVD attack, the FudModule loads an embedded version of the Dell dbutil_2_3.system driver, which contains an insufficient access control vulnerability (CVE-2021-21551) enabling the malware to escalate privileges. The FudModule extracts the driver files to C:\Windows\System32\drivers\ under a name randomly chosen from the following list:

- circlassmgr.sys
- dmvscmgr.sys
- hidirmgr.sys
- isapnpmgr.sys
- mspqmmgr.sys
- umpassmgr.sys

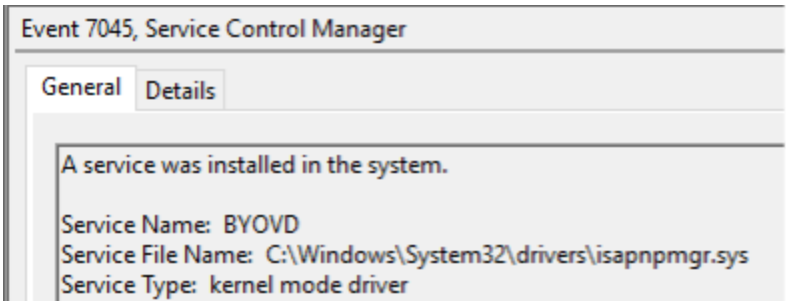
X-Force was unable to locate any legitimate driver installations associated with the filenames used by the FudModule so defenders may be able to detect the malicious driver installation through monitoring file activity where the filenames listed above are written to %windr%\system32\drivers.

Static Detection Opportunities Within Event Log Data

The main functionality of the FudModule is directed towards the Windows kernel space however when first loaded, the malware is operating within user mode meaning it cannot interact with kernel memory directly. To overcome this privilege issue, the FudModule executes the BYOVD attack leveraging the previously mentioned Dell driver. The Dell driver is installed as a kernel mode driver which combined with the CVE-2-1-2155 vulnerability enables the malware to the ability to escalate privileges to allow for writing to kernel memory.

In Microsoft Windows, when a kernel mode driver is installed a System Event Log Event ID 7045 is generated. System event log 7045 entries contain the name of the driver, the full path to the driver file, and the type of driver that was installed (either kernel or user mode). Defenders can detect the installation of the vulnerable driver as part of the FudModule by collecting Windows Event System Log entries where the Service Type is set to “kernel mode driver” and the Service File Name contains one of the following filenames:

- circlassmgr.sys
- dmvscmgr.sys
- hidirmgr.sys
- isapnpmgr.sys
- mspqmmgr.sys
- umpassmgr.sys



Static Detection Opportunities Within Setup API Log Data

The following detection opportunity is not specific to the FudModule. Throughout the research performed by X-Force on malicious use of the Dell DBUtil driver, X-Force identified additional malware and tools that leverage the Dell DBUtil BYOVD attack and identified the following detection point.

When the Dell Client Platform DBUtil driver is installed, the Microsoft Windows Plug and Play Devices module initiates the driver installation process. This process is captured within the SetupAPI log file located in %windir%\inf\setupapi.dev.log. The SetupAPI log was introduced in Windows Vista and contains information about device and driver installations including the initiating process, driver paths, and troubleshooting data.

Analysis of legitimate and malicious installations of the Dell DBUtil driver indicated that regardless of the implementation, the Device Install in the SetupAPI log will register the device as “ROOT\DBUtilDrv2”. Within the SetupAPI log file, defenders may be able to detect malicious installation of the DBUtil driver by collecting and parsing the SetupAPI log file for “ROOT\DBUtilDrv2” where the associated command and INF path do not match with the legitimate Dell Driver Install data shown below.

```
[Device Install (UpdateDriverForPlugAndPlayDevices) - ROOT\DBUtilDrv2]
Section start 2023/03/02 10:14:09.608
 cmd: "C:\Users\jdwyer\Downloads\DellTpm2.0_Fw1.3.2.8_V3_64.exe"
ndv: INF path: C:\Users\jdwyer\AppData\Local\Temp\DBUtilDrv2.inf
ndv: Install flags: 0x00000001
ndv: {Update Device Driver - ROOT\DELLUTILS\0000}
ndv:      Search options: 0x00000080
ndv:      Searching single INF 'C:\Users\jdwyer\AppData\Local\Temp\DBUtilDrv2.inf'
```

Legitimate DBUtil Install SetupAPI Data

```
>>>  [Device Install (UpdateDriverForPlugAndPlayDevices) - ROOT\DBUtilDrv2]
>>>  Section start 2023/03/03 14:45:28.652
      cmd: vulndrvtest.exe  -e 0 -p 728 -d c:\drvrtmp\
     ndv: INF path: c:\drvrtmp\\dbutildrv2.inf
     ndv: Install flags: 0x00000005
     ndv: {Update Device Driver - ROOT\DELLUTILS\0000}
     ndv:      Search options: 0x00000080
     ndv:      Searching single INF 'c:\drvrtmp\dbutildrv2.inf'
```

Malicious DBUtil Install SetupAPI Data

Static Detection Opportunity Through IOCs

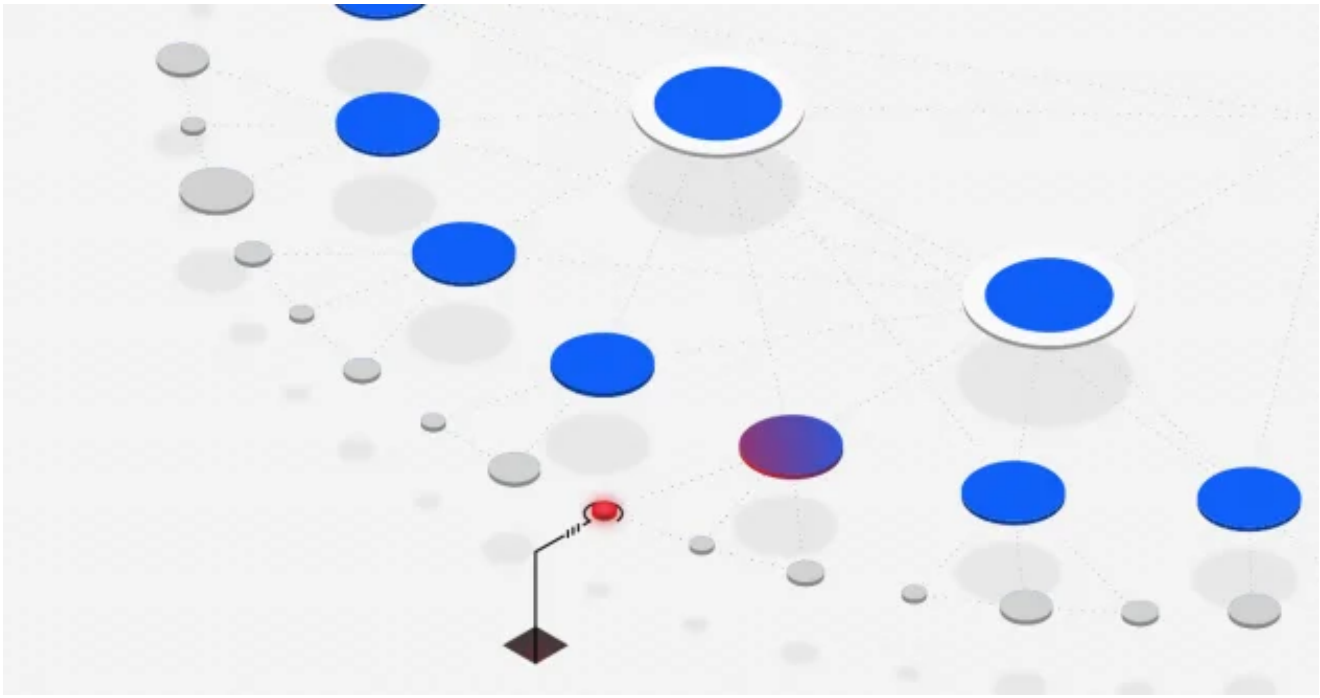
FudModule.dll File Hash:
97C78020EEDFCD5611872AD7C57F812B069529E96107B9A33B4DA7BC967BF38F

Dbutil_2_3.sys File Hash:
0296E2CE999E67C76352613A718E11516FE1B0EFC3FFDB8918FC999DD76A73A5

John Dwyer
Head of Research, IBM Security X-Force

John (@TactiKoolSec) is the Head of Research for the IBM Security X-Force where he leads research efforts to understand and model adversary operations, devel...

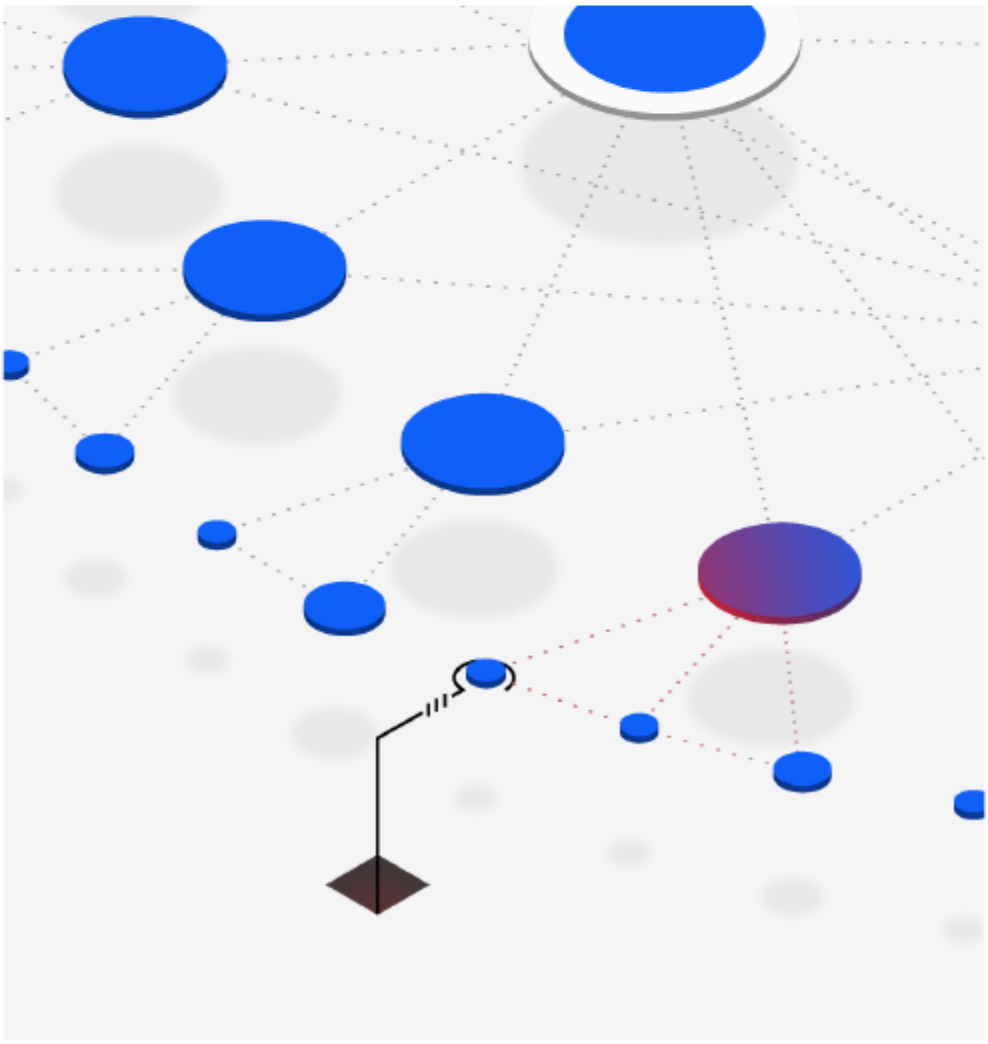
POPULAR



Intelligence & Analytics February 21, 2023

Backdoor Deployment and Ransomware: Top Threats Identified in X-Force Threat Intelligence Index 2023

4 min read - Discover how threat actors are waging attacks and how to proactively protect your organization with top findings from the 2023 X-Force Threat Intelligence Index.



IBM

IBM Security X-Force Threat Intelligence Index: Explore the

top threats of 2022.

Read the report →