x86matthew - SetTcpEntry6 - A custom SetTcpEntry implementation for IPv6

Windows exports a function called SetTcpEntry which can be used to terminate an active connection in the TCP table. This function only supports IPv4 connections, and no equivalent function for IPv6 currently exists. Even Microsoft's own TCPView utility doesn't allow IPv6 connections to be closed.

I have reverse-engineered the original SetTcpEntry function and have successfully created a custom SetTcpEntry6 function. This code works on all versions of Windows with the "Next Generation TCP/IP Stack" (Vista onwards). Windows XP doesn't fully support IPv6 to begin with, so these functions will work as expected under all realistic circumstances.

The SetTcpEntry function calls NsiSetAllParameters with an undocumented data structure. I have reverse-engineered this structure and labelled it KillTcpSocketData_V4. My reverse-engineered SetTcpEntry (IPv4) function is below:

```
BYTE bGlobal_NPI_MS_TCP_MODULEID[] = { 0x18, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x03, 0x4A, 0x00, 0xEB, 0x1A, 0x9B, 0xD4, 0x11, 0x91, 0x23, 0x00, 0x50, 0x04, 0x77, 0x59, 0xBC };

struct KillTcpSocketData_V4
{
WORD wLocalAddressFamily;
WORD wLocalPort;
DWORD dwLocalAddr;
BYTE bReserved1[20];

WORD wRemoteAddressFamily;
WORD wRemotePort;
DWORD dwRemoteAddr;
BYTE bReserved2[20];
};

DWORD KillTcpSocket_V4(MIB_TCPROW_OWNER_PID *pTcpRow)
{
HMODULE hNsiModule = NULL;
DWORD (WINAPI *pNsiSetAllParameters)(DWORD dwStatic, DWORD dwActionCode, LPVOID NPI_MS_MODULEID, DWORD dwIoMainCode, LPVOID lpNetInfoBuffer, DWORD SizeofNetInfoBuffer, LPVOID lpMetricBuffer, DWORD SizeofMetricBuffer) = NULL;
KillTcpSocketData_V4 KillTcpSocketData;
```

```
MIB_TCPROW_OWNER_PID SetTcpEntryTcpRow;

// load nsi.dll module (vista onwards)
hNsiModule = LoadLibrary("nsi.dll");
if(hNsiModule != NULL)
{
// find NsiSetAllParameters function
pNsiSetAllParameters = (unsigned long (__stdcall *)(unsigned long,unsigned long,void
*,unsigned long,void *,unsigned long,void *,unsigned long))GetProcAddress(hNsiModule,
"NsiSetAllParameters");
if(pNsiSetAllParameters == NULL)
{
return 1;
}
}

if(pNsiSetAllParameters == NULL)
{
// NsiSetAllParameters not found - call original SetTcpEntry function
memcpy((void*)&SetTcpEntryTcpRow;, (void*)pTcpRow, sizeof(SetTcpEntryTcpRow));
SetTcpEntryTcpRow.dwState = MIB_TCP_STATE_DELETE_TCB;
if(SetTcpEntry(&SetTcpEntryTcpRow;) != 0)
{
return 1;
}
}
else
{
// set socket data
memset((void*)&KillTcpSocketData;, 0, sizeof(KillTcpSocketData));
KillTcpSocketData.wLocalAddressFamily = AF_INET;
KillTcpSocketData.wLocalPort = (WORD)pTcpRow->dwLocalPort;
KillTcpSocketData.dwLocalAddr = pTcpRow->dwLocalAddr;
KillTcpSocketData.wRemoteAddressFamily = AF_INET;
KillTcpSocketData.wRemotePort = (WORD)pTcpRow->dwRemotePort;
KillTcpSocketData.dwRemoteAddr = pTcpRow->dwRemoteAddr;

// kill socket
if(pNsiSetAllParameters(1, 2, (LPVOID)bGlobal_NPI_MS_TCP_MODULEID, 16,
(LPVOID)&KillTcpSocketData;, sizeof(KillTcpSocketData), 0, 0) != 0)
{
return 1;
}
}
```

```
return 0;
}
```

The function above isn't really necessary since the existing SetTcpEntry function works with IPv4 sockets on all versions of Windows, but it serves as a starting point for creating an IPv6 implementation.

After spending some time looking at other IPv6-related functions in nsi.dll, I came up with the following code:

```
BYTE bGlobal_NPI_MS_TCP_MODULEID[] = { 0x18, 0x00, 0x00, 0x00, 0x01, 0x00,
0x00, 0x00, 0x03, 0x4A, 0x00, 0xEB, 0x1A, 0x9B, 0xD4, 0x11, 0x91, 0x23, 0x00, 0x50,
0x04, 0x77, 0x59, 0xBC };

struct KillTcpSocketData_V6
{
WORD wLocalAddressFamily;
WORD wLocalPort;
BYTE bReserved1[4];
BYTE bLocalAddr[16];
DWORD dwLocalScopeID;

WORD wRemoteAddressFamily;
WORD wRemotePort;
BYTE bReserved2[4];
BYTE bRemoteAddr[16];
DWORD dwRemoteScopeID;
};

DWORD KillTcpSocket_V6(MIB_TCP6ROW_OWNER_PID *pTcpRow)
{
HMODULE hNsiModule = NULL;
DWORD (WINAPI *pNsiSetAllParameters)(DWORD dwStatic, DWORD dwActionCode,
LPVOID NPI_MS_MODULEID, DWORD dwIoMainCode, LPVOID lpNetInfoBuffer,
DWORD SizeofNetInfoBuffer, LPVOID lpMetricBuffer, DWORD SizeofMetricBuffer) =
NULL;
KillTcpSocketData_V6 KillTcpSocketData;

// load nsi.dll module (vista onwards)
hNsiModule = LoadLibrary("nsi.dll");
if(hNsiModule != NULL)
{
// find NsiSetAllParameters function
pNsiSetAllParameters = (unsigned long (__stdcall *)(unsigned long,unsigned long,void
```

```
*,unsigned long,void *,unsigned long,void *,unsigned long))GetProcAddress(hNsiModule,
"NsiSetAllParameters");
if(pNsiSetAllParameters == NULL)
{
return 1;
}
}

if(pNsiSetAllParameters == NULL)
{
// NsiSetAllParameters not found (win XP or earlier - ipv6 not supported)
return 1;
}

// set socket data
memset((void*)&KillTcpSocketData;, 0, sizeof(KillTcpSocketData));
KillTcpSocketData.wLocalAddressFamily = AF_INET6;
KillTcpSocketData.wLocalPort = (WORD)pTcpRow->dwLocalPort;
memcpy((void*)KillTcpSocketData.bLocalAddr, (void*)pTcpRow->ucLocalAddr,
sizeof(KillTcpSocketData.bLocalAddr));
KillTcpSocketData.dwLocalScopeID = pTcpRow->dwLocalScopeId;
KillTcpSocketData.wRemoteAddressFamily = AF_INET6;
KillTcpSocketData.wRemotePort = (WORD)pTcpRow->dwRemotePort;
memcpy((void*)KillTcpSocketData.bRemoteAddr, (void*)pTcpRow->ucRemoteAddr,
sizeof(KillTcpSocketData.bRemoteAddr));
KillTcpSocketData.dwRemoteScopeID = pTcpRow->dwRemoteScopeId;

// kill socket
if(pNsiSetAllParameters(1, 2, (LPVOID)bGlobal_NPI_MS_TCP_MODULEID, 16,
(LPVOID)&KillTcpSocketData;, sizeof(KillTcpSocketData), 0, 0) != 0)
{
return 1;
}

return 0;
}
```

I have tested my KillTcpSocket_V6 function on all versions of Windows from Vista up to the latest version of Windows 10 with successful results.