Relate

- Make a Windbg By Yourself
- Make yourself more powerful!
- Learn to make yourself happy
- Windbg
- WINDBG
- Looking for work, how to make yourself stand out in the interview?
- Looking for a good job?12 elements resume to make yourself stand out
- Make you more love their top ten super self-suggestion (10 Powerful Self-affirmations to Love Yourself More)
- Wrestle with WinDBG
- Using windbg

Make a Windbg By Yourself

Tag: **Author:** kingston123 **Date:** 2012-11-14

The Windbg substance of the work is connected to the Windows kernel debugger windows has kernel debugging mechanism integrated into the kernel, only need to connect the windows kernel debugger interface, do not need to think like softice mounts a large number of interrupt handling process will be able to achieve the kernel debugger,
Details on this, please refer to the Windows kernel debugger principle Analysis

Ever heard of themida Daniel windbg is just a demo, fully able to write better, even not a cow, it is not possible to write good things must first try to be imitation, not yet capable of reproductions, not to talk about beyond the

windbg kernel debugging some of the main features packaged into the dbgeng.dll by debugcreate create a CLSID COM + interface to achieve the debug control we can come up with the ida analysis windbg.exe

```
signed int __ stdcall CreateUiInterfaces (int a2, PCWSTR RemoteOptions)
{
............

ReleaseUiInterfaces ();
if (a2)
{
v3 = DebugConnectWide (RemoteOptions, & _GUID_27fe5639_8407_4f47_8364_ee118fb08ac8, & g_UiClientBase);

.............
}
else
{
v6 = DebugCreate (& _GUID_27fe5639_8407_4f47_8364_ee118fb08ac8, & g_UiClientBase);
```

As shown DebugConnectWide connected to the remote debug module

In fact, we directly use the dbgeng.h + dbgeng.lib, can be completed in windbg all the features
dbgeng.h file C :/ Program Files / Debugging Tools for Windows (x86) / sdk / inc
This file size is 521KB, including all windbg debugapi
Windbg installation must choose to install the sdk, otherwise I might not have this file

```
/ / RemoteOptions specifies connection types and
/ / Their parameters. Supported strings are:
/ / Npipe: Server = <Machine>, Pipe = <Pipe name>
/ / Tcp: Server = <Machine>, Port = <IP port>
```

```
STDAPI
DebugConnectWide (
__in PCWSTR RemoteOptions,
__in REFIID InterfaceId,
__out PVOID * Interface
);

STDAPI
DebugCreate (
__in REFIID InterfaceId,
__out PVOID * Interface
);

STDMETHOD (GetThreadContext) (
THIS_
__out_bcount (ContextSize) / * align_is (16) * / PVOID Context,
__in ULONG ContextSize
) PURE;
STDMETHOD (SetThreadContext) (
THIS_
__in_bcount (ContextSize) / * align_is (16) * / PVOID Context,
__in ULONG ContextSize
) PURE;
```

Above api Statement, we can directly use these APIs to complete commissioning work

Although we can achieve the Debugger, but there is still some trouble an open source PyDbgEng of

Can get a lot of information from the above site inside us look at the introduction
PyDbgEng is a Python Wrapper For Microsoft Debug Engine.
kernel mode debugging
x86, x64 support
Wrapper for DebugCreate () API which creates IDebugClient COM interface.

Easy access to IDebugClient COM interface
Easy access to all other DbgEng COM interfaces via IDebugClient.QueryInterface ()
Easy access to all DbgEng structs and enums.
Receive DbgEng events. Currently supported: IDebugEventCallbacks, IDebugOutputCallbacks

The good completely encapsulated IDebugClient COM interface so that we can directly use
py code has the advantage is that you can write a test program the fastest

We arbitrarily select a piece of code

```
event_handler = DbgEventHandler ()
dbg = PyDbgEng.KernelAttacher (connection_string = connection_string, /
set_initial_bp = True, /
event_callbacks_sink = event_handler, /
output_callbacks_sink = event_handler, /
symbols_path = "SRV * ")
--------------------------
class DbgEventHandler (PyDbgEng.IDebugOutputCallbacksSink, PyDbgEng.IDebugEventCallbacksSink):

def LoadModule (self, dbg, ImageFileHandle, BaseOffset, ModuleSize, ModuleName, ImageName, CheckSum,
TimeDateStamp):
sys.stdout.write ("LoadModule: ImageName = /"% s / "/ n"% ImageName)
return PyDbgEng.DbgEng.DEBUG_STATUS_NO_CHANGE is
---------------------------
```

KernelAttacher connection, set the event, then you can deal directly with the event, it is not very convenient

We look at a r3debuger

```
    event_handler = DbgEventHandler ()
    dbg = PyDbgEng.ProcessCreator (command_line = filename, /
    follow_forks = true, /
    event_callbacks_sink = event_handler, /
    output_callbacks_sink = event_handler, /
    symbols_path = "SRV * [url] [/ url]")

    to def NtCreateThread_at_entry (dbg, args):
    sys.stdout.write ("NtCreateThread () called with following call stack :/ n")
    stack_frames = dbg.get_stack_trace (FRAMES_COUNT)
    for i in range (FRAMES_COUNT):
    eip = stack_frames [i]. InstructionOffset
    if (eip == 0):
    break
    func_symbol = dbg.get_symbol (eip)
    sys.stdout.write ("[% d]% s / n"% (i, func_symbol))
    sys.stdout.write ("/ n")
```

Code is substantially similar, ProcessCreator selected process, and then carry out various working

We can find, using PyDbgEng + dbgeng.dll allows us to write your own debugger
's Task easy, only need to complete eventhandle event handler to make any changes without the need for complex processing mechanism can achieve what we want

Mentioned in the text tool
1.windbg
Install Debugging Tools for Windows 32-bit Version

Install Debugging Tools for Windows 64-bit Versions

2.PyDbgEng

lin spots recommended tool is too strong