Flier's Sky

Labels

debugging (3)
google chrome (1)
look'n'stop (1)
plugin (1)
security (3)
software (1)
virtualization (3)
wireshark (1)

Links

Flier's Sky - My Chinese Blog
Flier's Sky - My Chinese Tech Blog
PyDbgExt - Python extension for WinDbg
PyV8 - Python wrapper for V8 Javascript Engine
python-google-url/ - A python wrapper for google-url project

Blog Archive

► 2008 (3)
▼ 2007 (6)
► August (1)
► July (1)
▼ May (4)
Write a debugger in 5 minutes with PyDbgEng
Access the kernel space with PyDbgEng
Dump Windows Service Table in WinDbg
An alternative open source virtualization solution...

Flier Lu
Beijing, Beijing, China
View my complete profile

LineBuzz

Monday, May 28, 2007

Write a debugger in 5 minutes with PyDbgEng

The debug mechanism of PyDbgEng is same to other Win32 debugger, just create or attach to a debuggee process and call WaitForEvent to process the debug events, such as create process, load module etc.

```
#!/usr/bin/env python
import sys
from PyDbgEng import *

c = DebugClient()

c.CreateProcess("ftp.exe", createFlags=[CreateFlags.ATTACH_ONL
Y_THIS_PROCESS,
  CreateFlags.NEW_CONSOLE])

while c.Control.WaitForEvent():
pass
```

The previous code is a simple debugger

1. create a debug session with DebugClient()
2. create the debuggee process with CreateProcess
3. attach to the new process with CreateFlags.ATTACH_ONLY_THIS_PROCESS
4. create a new console window for debuggee with CreateFlags.NEW_CONSOLE

To get more debug events, we must add some debug event callback

```
def onCreateProcess(args):
print "CreateProc: %08x-%08x %s\t%s" % (
args.BaseOffset, args.BaseOffset+args.ModuleSize,
args.ModuleName, args.ImageName)

def onExitProcess(args):
print "ExitProcess %d" % args.ExitCode

def onCreateThread(args):
print "CreateThread %x %08x %08x" % (args.Handle, args.DataOff
set, args.StartOffset)

def onExitThread(args):
print "ExitThread %d" % args.ExitCode

def onLoadModule(args):
print "ModLoad: %08x-%08x %s\t%s" % (
args.BaseOffset, args.BaseOffset+args.ModuleSize,
args.ModuleName, args.ImageName)

c.EventCallbacks.CreateProcess = onCreateProcess
c.EventCallbacks.ExitProcess = onExitProcess
c.EventCallbacks.CreateThread = onCreateThread
c.EventCallbacks.ExitThread = onExitThread
c.EventCallbacks.LoadModule = onLoadModule
```

```
c.EventCallbacks.Attach()

c.CreateProcess(...)
```

Now, we will receive 5 kinds of debug events, which allow use show detail information, or do some action, such as add breakpoint, etc. After setting the callback and attach them to debugger, we can got events, like:

CreateProc: 01000000-01012000 ftp ftp.exe
ModLoad: 7c930000-7ca00000 ntdll ntdll.dll
ExitProcess 0

Other kinds of events are about the status or state changing for debug session, debuggee and symbol, we also can use callback to process them

```
def onSessionStatus(args):
print "SessionStatus: %s" % (str(args.Status))

def onChangeEngineState(args):
sys.stdout.write("EngineState: %s " % str(args.State))

if EngineState.EXECUTION_STATUS == args.State:
 print ExecutionStatus.values[args.Argument & 0xf]
else:
 print "%x" % args.Argument

c.EventCallbacks.SessionStatus = onSessionStatus
c.EventCallbacks.ChangeEngineState = onChangeEngineState
```

These events will allow you watch the order of state changing, like

EngineState: SYSTEMS 0
EngineState: EXECUTION_STATUS NO_CHANGE
EngineState: EXTENSIONS 0
SessionStatus: ACTIVE
EngineState: EXECUTION_STATUS BREAK
EngineState: CURRENT_THREAD 0
CreateProc: 01000000-01012000 ftp ftp.exe
EngineState: EXECUTION_STATUS BREAK
EngineState: EXECUTION_STATUS BREAK
EngineState: CURRENT_THREAD 0
ModLoad: 7c930000-7ca00000 ntdll ntdll.dll
EngineState: EXECUTION_STATUS BREAK
EngineState: EXECUTION_STATUS BREAK
EngineState: CURRENT_THREAD 0

To act as a complete debugger, we add and process breakpoint for some predefined function

```
def onLoadModule(args):
print "ModLoad: %08x-%08x %s\t%s" % (
  args.BaseOffset, args.BaseOffset+args.ModuleSize,
  args.ModuleName, args.ImageName)

if "WS2_32" == args.ModuleName:
  bp = c.Control.AddBreakpoint(flags=[BreakpointFlag.ENABLED],
      offset=c.Symbols.GetOffsetByName("WS2_32!socket"))

  symbol = c.Symbols.GetNameByOffset(bp.Offset)
  print "Add Breakpoint: %s %d @ %08x %s:%d" % (str(bp.Type[0]
), bp.Id, bp.Offset, symbol[0], symbol[1])

def onBreakpoint(args):
bp = args.Breakpoint

symbol = c.Symbols.GetNameByOffset(bp.Offset)
print "Hit Breakpoint: %s %d @ %08x %s:%d" % (str(bp.Type[0]),
```

```
 bp.Id, bp.Offset, symbol[0], symbol[1])

return ExecutionStatus.BREAK

c.EventCallbacks.Breakpoint = onBreakpoint
```

After the WS2_32 module was loaded, we use DebugControl.AddBreakpoint method to create a new code break for WS2_32!socket, which will be called after ftp.exe started. So we add onBreakpoint callback function to show which breakpoint was hit.

```
ModLoad: 71b60000-71b77000 WS2_32
C:\WINDOWS\system32\WS2_32.dll
EngineState: BREAKPOINTS 0
EngineState: BREAKPOINTS 0
EngineState: BREAKPOINTS 0
Add Breakpoint: CODE 0 @ 71b6410c WS2_32!socket:0
...
EngineState: CURRENT_THREAD 0
Hit Breakpoint: CODE 0 @ 71b6410c WS2_32!socket:0
EngineState: EXECUTION_STATUS BREAK
EngineState: EXECUTION_STATUS GO_HANDLED
Change engine state to GO
EngineState: EXECUTION_STATUS GO_HANDLED
EngineState: EXECUTION_STATUS GO
```

Besides these expected events, we need another callback to process the exception.

```
def onException(args):
 symbol = c.Symbols.GetNameByOffset(args.Address)
 sys.stdout.write("Exception: %08x %08x %s:%d" % (args.Code, a
rgs.Address, symbol[0], symbol[1]))

 if args.IsFirstChance:
   print " first"
 else:
   print " second"

 for frame in c.Control.GetStackFrames():
   symbol = c.Symbols.GetNameByOffset(frame.InstructionOffset)
   print " %04d %08x %s:%d" % (frame.FrameNumber, frame.Instru
ctionOffset, symbol[0], symbol[1])

 print c.Control.Breakpoints

c.EventCallbacks.Exception = onException
```

The callback will log the exception information, and dump the caller stack with DebugControl.GetStackFrames() method, like

```
Exception: 000006ba 7c80bee7 kernel32!RaiseException:83
first
0000 7c80bee7 kernel32!RaiseException:83
0001 77c31e37 RPCRT4!RpcpRaiseException:36
0002 77c32042 RPCRT4!NdrGetBuffer:70
0003 77cb30e4 RPCRT4!NdrClientCall2:407
0004 76e35039 DNSAPI!R_ResolverQuery:28
0005 76e34f59 DNSAPI!Query_PrivateExW:391
0006 76e3505f DNSAPI!DnsQuery_W:58
0007 71a83f8e MSWSOCK!SaBlob_Query:45
...
0023 010045c5 ftp!main:1665
0024 01006ee0 ftp!mainCRTStartup:303
0025 7c82f23b kernel32!BaseProcessStart:35
```

Finally, we add a try...except to protect the WaitForEvent method, because some situation will raise exception

```
try:
while c.Control.WaitForEvent():
    c.Control.ExecutionStatus = ExecutionStatus.GO_HANDLED
    print "Change engine state to %s" % c.Control.ExecutionSta
tus
except:
if ExecutionStatus.NO_DEBUGGEE != c.Control.ExecutionStatus:
    print "Unexpected error:", sys.exc_info()[0]
    raise
```

Now, its work, with less than one hundred code lines, and can be expand easy :)

Submit by Flier Lu @ 12:14:00 AM Lables: debugging

No comments:

Post a Comment

指向此文章的链接

Create a Link

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)