G+1 0 Liên kết khác Blog tiếp theo» Tạo Blog Đăng nhập

Flier's Sky

Saturday, May 26, 2007

Dump Windows Service Table in WinDbg

buri write a great article <Windows Service Table Dumper for WinDbg> show how to use the built-in script language in WinDbg to do a real job: dump the windows service table. But this script is short of readability, because the build-in script in WinDbg is very strange like its command design.
So, why we can't implement it more easy and readable, base on a friendly python script through PyDbgExt, my python extension for WinDbg :)

First we need define a script module, such as dumpServiceTable.py, which includes a function dumpServiceTable to dump that table, and import the dependence modules

```
from PyDbgEng import *
from struct import *

c = DebugClient.Current
s = c.Symbols
v = c.DataSpaces.Virtual
```

Next, we got the common base object, such as DebugClient.Current which is the current debug session in windbg; Symbols and DataSpaces.Virtual will support us query the debug symbol and read/write the virtual address space.

```
def getSymbol(name):
return s.GetSymbols(name).popitem()[1][0]

def getSymbol(offset):
return s.GetSymbols(offset).popitem()[1][0]

def readDWORD(offset):
return unpack_from("L", v.Read(offset, 4))[0]
```

To make the code more readable, we define some utility functions: getSymbol can get the symbol object with its name or offset; readDWORD read unsigned long from the offset. According to the result type of VirtualDataSpace.Read function is a buffer object, we need use unpack_from function to decode the buffer.

```
def dumpServiceTable():
KiServiceTable = getSymbol("nt!KiServiceTable")
KiServiceLimit = getSymbol("nt!KiServiceLimit")

idx = 0

for addr in v.ReadPointers(KiServiceTable.Offset,
```

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

```
readDWORD(KiServiceLimit.Offset)):
try:
symbol = getSymbol(addr)

symbolName = "%s!%s" % (symbol.Module.ModuleName,
symbol.Name)
except:
symbolName = ""

print "%03d %08x %s" % (idx, addr & 0xFFFFFFFF,
symbolName)

idx = idx + 1
```

The last part of code read and dump the service table:

1. get the symbol object of nt!KiServiceTable and nt!KiServiceLimit
2. read a group of pointers from the begin of table
3. try to get the symbol object for every entry in table
4. if the symbol exists, dump it's address, module and name
5. if the symbol nonexists, just show warning. we can provide more information about this in future

Finally, we load the script to windbg and execute it :)

```
lkd> .extpath+ D:\Study\Win32\PyDbgExt\Binary\debug
Extension search path is:
...;D:\Study\Win32\PyDbgExt\Binary\debug
lkd> .load PyDbgExt
lkd> .chain
Extension DLL search Path:
...
Extension DLL chain:
PyDbgExt: API 1.0.0, built Sat May 26 02:17:49 2007
[path: D:\Study\Win32\PyDbgExt\Binary\debug\PyDbgExt.dll]
dbghelp: image 6.7.0005.0, API 6.0.6, built Fri Mar 30
02:08:09 2007
[path: D:\MS\Debugging Tools for Windows\dbghelp.dll]
...
lkd> !import dumpServiceTable
Import succeeded.
lkd> !eval dumpServiceTable.dumpServiceTable()
000 8092023a nt!NtAcceptConnectPort
001 8096b71e nt!NtAccessCheck
002 8096f9be nt!NtAccessCheckAndAuditAlarm
...
032 808b9810 nt!NtCompressKey
033 f4bed0d2
034 8088d0c8 nt!NtContinue
...
```

If there some wrong in script, just edit it and reload it with python build-in function

```
lkd> !eval reload(dumpServiceTable)
```

Enjoy it :)

Submit by Flier Lu @ 3:01:00 AM Lables: debugging, security

1 comment:

id said...

This is fantastic!

October 28, 2010 at 4:39 PM

Post a Comment

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)