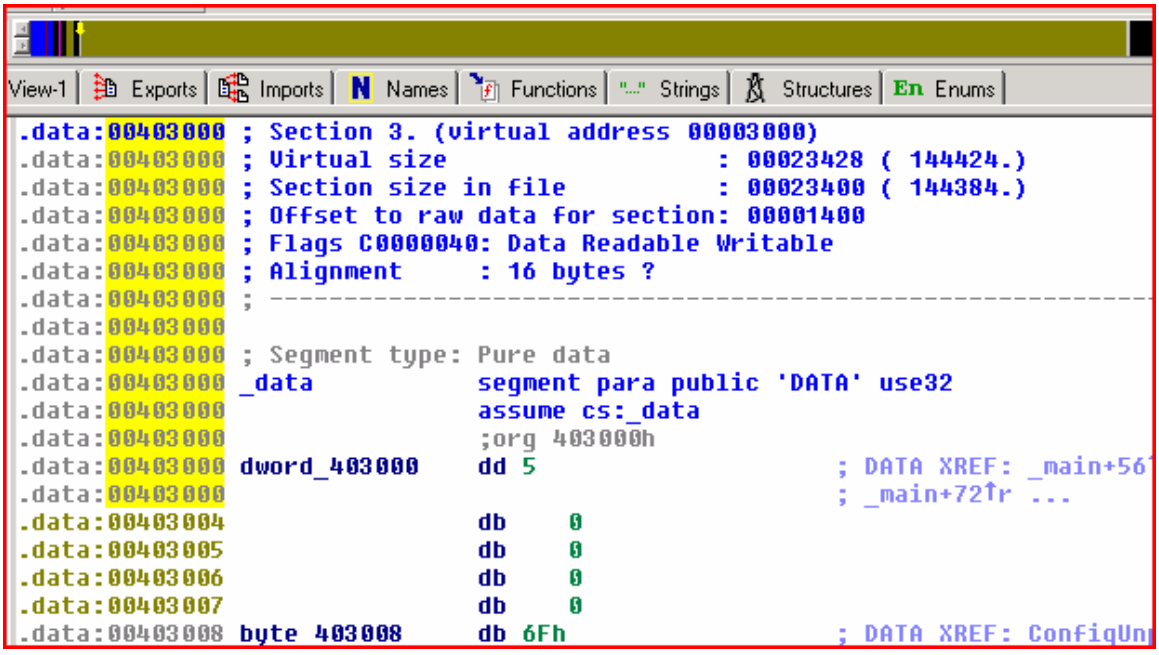
Extracting Xor-Encoded Files From Malware With IDAPython

I. Problem statement:

This specimen contains a 144424 byte .data section, 99.994% of which is left unexplored by IDA. Without using a debugger or otherwise running the code, how can one determine the purpose of these bytes and turn it into something useful.



II. Research:

A few steps into the start of the code, there exists the following function that copies 5 offsets into some variables, all of which point into the unexplored .data section.

```
.text:00401000 sub_401000 proc near  ; CODE XREF: _func+4Ap
.text:00401000     mov     lpBuffer, offset byte_403008
.text:0040100A     mov     dword_4263F0, offset unk_403C08
.text:00401014     mov     dword_4263F4, offset unk_404008
.text:0040101E     mov     dword_4263F8, offset unk_416608
.text:00401028     mov     dword_4263FC, offset unk_416648
.text:00401032     retn
.text:00401032 sub_401000 endp
```

Shortly thereafter, there is a call to the following function, once for each variable. The parameters are the offsets created above and a dword pointer, which turns out to be the size in bytes of the xor-encoded file.

```
.text:00401033 ; ¦¦¦¦ S U B R O U T I N E ¦¦¦¦
.text:00401033
.text:00401033
.text:00401033 xor       proc near  ; CODE XREF: _func+6Cp
.text:00401033
.text:00401033 arg_0     = dword ptr  4
.text:00401033 arg_4     = dword ptr  8
.text:00401033
.text:00401033           xor     ecx, ecx
.text:00401035           cmp     [esp+arg_4], ecx
.text:00401039           jle     short loc_40104B
.text:0040103B
.text:0040103B loc_40103B:   ; CODE XREF: xor+16j
.text:0040103B           mov     eax, [esp+arg_0]
.text:0040103F           add     eax, ecx
.text:00401041           xor     byte ptr [eax], 22h
.text:00401044           inc     ecx
.text:00401045           cmp     ecx, [esp+arg_4]
.text:00401049           jl      short loc_40103B
.text:0040104B
.text:0040104B loc_40104B:   ; CODE XREF: xor+6j
.text:0040104B           xor     eax, eax
.text:0040104D           retn
.text:0040104D xor       endp
```

III. Solution

This is really all the information needed to begin extracting the xor-encoded files from this piece of malware. Since the recent Word 0-day also had really simple xor-encoded files embedded (using 0xFF), it would be nice to produce something re-usable. The following IDAPython script has a DoGeneric() function to accept user input for output file name, start address, size, and xor key. The DoSpecific() function exists to automatically extract the 5 files from this particular specimen.

```
# extract (embedded) xor-encoded files from malware

import struct

def DoExtract(fn,addr,size,key):
  out=file(fn,'wb')
  for i in range(size-1):
    data=struct.pack('h',get_byte(addr+i)^key)
    if i>0:
      out.seek(-1,1)
    out.write(data)
  out.close()

def DoGeneric():
  fn=AskFile(1,"c:\output.exe","Enter output file: ")
  ea=askaddr(get_screen_ea(),"Enter start address: ")
  size=asklong(0xC00,"Enter length: ")
  key=asklong(0x22,"Enter xor key: ")
  if fn==None or ea==None or size==None or size==0:
    print "Error!\n"
  else:
```

```
    DoExtract(fn,ea,size,key)
  exit(0)

def DoSpecific():
  DoExtract("c:\\files\\1.out",0x00403008,0xC00,0x22)
  DoExtract("c:\\files\\2.out",0x00403C08,0x400,0x22)
  DoExtract("c:\\files\\3.out",0x00404008,0x12600,0x22)
  DoExtract("c:\\files\\4.out",0x00416608,0x40,0x22)
  DoExtract("c:\\files\\5.out",0x00416648,0xFC00,0x22)

#DoGeneric()
DoSpecific()
```

IV. Conclusion

The files can now be retrieved from disk and examined individually, without having to run the code. This is good because during execution of the program, it will extract, run, and immediately delete two of the files.

V. Caveats

Python's write() method is string-based, creating the need to use struct.pack() to output binary data. Even then, each byte is accommodated with a NUL when the write occurs. For the script to work properly, it must do a check and use seek() to overwrite the previous write's NUL before writing the next byte. This works fine, but can be a lot cleaner.

VI. Resources

IDA http://www.datarescue.com
IDAPython http://d-dome.net/idapython