Using SABRE BinDiff v1.6 for Malware analysis

With more and more malware surfacing every week, and the trend towards malware "families", AV analysts are faced with a flood of code to analyze and disassemble. The pieces of malware keep getting larger and more complex. Many include SMTP servers and other sophisticated functionality. The various members of the SoBig family were of the size of small applications and shared a significant amount of code. Almost all worms or virii spawn a number of variants and mutations very quickly. This situation is aggravated by some malware being spread in source format for easy adaption to the needs of the attacker. In this situation, the authors of malware try to exploit the asymmetry in the workload between changing/recompiling source and analyzing/disassembling the compiled executable. Using SABRE BinDiff v1.6, the workload involved in analyzing multiple variants of the same piece of malware can be drastically reduced.

Function names and comments that were created during the analysis of one variant can be ported to other variants quickly and easily. For the purposes of this paper, we will utilize two IDA Pro databases – BagleX.idb, which is an untouched disassembly of Bagle/X, and BagleW.idb, which is a heavily commented disassembly of Bagle/W.

The disassembly of Bagle/W contains different sort of comments – repeatable comments on many functions, extra comment lines before/after instructions, and per-instruction comments. All functions have meaningful names.

The disassembly of Bagle/X is an untouched IDA disassembly. It thus contains no comments except those created by IDA, no meaningful names, and no anterior/posterior comment lines. In the following few pages, this paper will show in a step-by-step manner how SABRE BinDiff v1.6 can be used to re-use the information gained from Bagle.W for the disassembly of Bagle/X.

```
.text:0040186B
.text:0040186B IMAGESTUFF__PrepareBitmap proc near     ; CODE XREF: IMAGESTUFF__GetBitmapFromString+31↓p
.text:0040186B
.text:0040186B Rect            = tagRECT ptr -14h
.text:0040186B hDC             = dword ptr -4
.text:0040186B lpString        = dword ptr  8
.text:0040186B arg_4_W         = dword ptr  0Ch
.text:0040186B arg_8_H         = dword ptr  10h
.text:0040186B
.text:0040186B                 push    ebp
.text:0040186C                 mov     ebp, esp
.text:0040186E                 add     esp, 0FFFFFFECh
.text:00401871                 push    ebx
.text:00401872                 mov     [ebp+Rect.left], 0
.text:00401879                 push    [ebp+arg_4_W]
.text:0040187C                 pop     [ebp+Rect.right]
.text:0040187F                 mov     [ebp+Rect.top], 0
.text:00401886                 push    [ebp+arg_8_H]
.text:00401889                 pop     [ebp+Rect.bottom]
.text:0040188C                 push    0               ; HDC
.text:0040188E                 call    CreateCompatibleDC
.text:0040188E creates a memory device context compatible with the application's current screen
.text:00401893                 mov     [ebp+hDC], eax
.text:00401896                 push    BITSPIXEL       ; int
.text:00401898                 push    [ebp+hDC]       ; HDC
.text:0040189B                 call    GetDeviceCaps
.text:0040189B retrieve the number of bits per pixel for current display
.text:0040189B (we need to pass it to CreateBitmap)
.text:004018A0                 push    0               ; void * -- no color data
.text:004018A2                 push    eax             ; UINT
.text:004018A3                 push    1               ; UINT -- 1 color plane for current image
.text:004018A5                 push    [ebp+arg_8_H]   ; int
.text:004018A8                 push    [ebp+arg_4_W]   ; int
.text:004018AB                 call    CreateBitmap
.text:004018AB result: a new bitmap in the device context
```

Illustration 1 The commented disassembly of Bagle/W

For this, the uncommented disassembly of Bagle/X needs to be open in IDA, and the disassembly of Bagle/W must not be opened by any other IDA instance. Hitting CTRL-5 pops up the SABRE BinDiff screen. We click on the "Configuration"-Button and are faced with the screen shown in Illustration 2. The "Temp Directory" has to point to an intermediate directory in which data needed for the porting of comments can be stored. The configuration shown below is the default configuration – the only changes that need to be done is checking the "Function Names" and "Comments" checkboxes in the "Port" group.

After clicking "OK", the "Diff Database against" button is clicked, and the file "BagleW.idb" is selected. The program will now process the two disassemblies, recognizing unchanged and slightly changed code pieces.

Once the processing is finished, three screens will appear in your IDA Pro Window: "Unmatched: Current IDB", "Unmatched: Other IDB" and the screen that is of primary interest to us: "Matched Functions". The screen consists of five columns: One that indicates whether the function in that row changed between the two disassemblies, and two columns indicating the address and name of the function in the two disassemblies.

In the present example, the functions with meaningful names can be seen on the right, whereas their equivalents without meaningful names can be seen on the left (see also Illustration 3). Out of the 236 functions in the Bagle/X sample, 223 are identical to functions in the disassembly of Bagle/W, and their names can be easily ported.

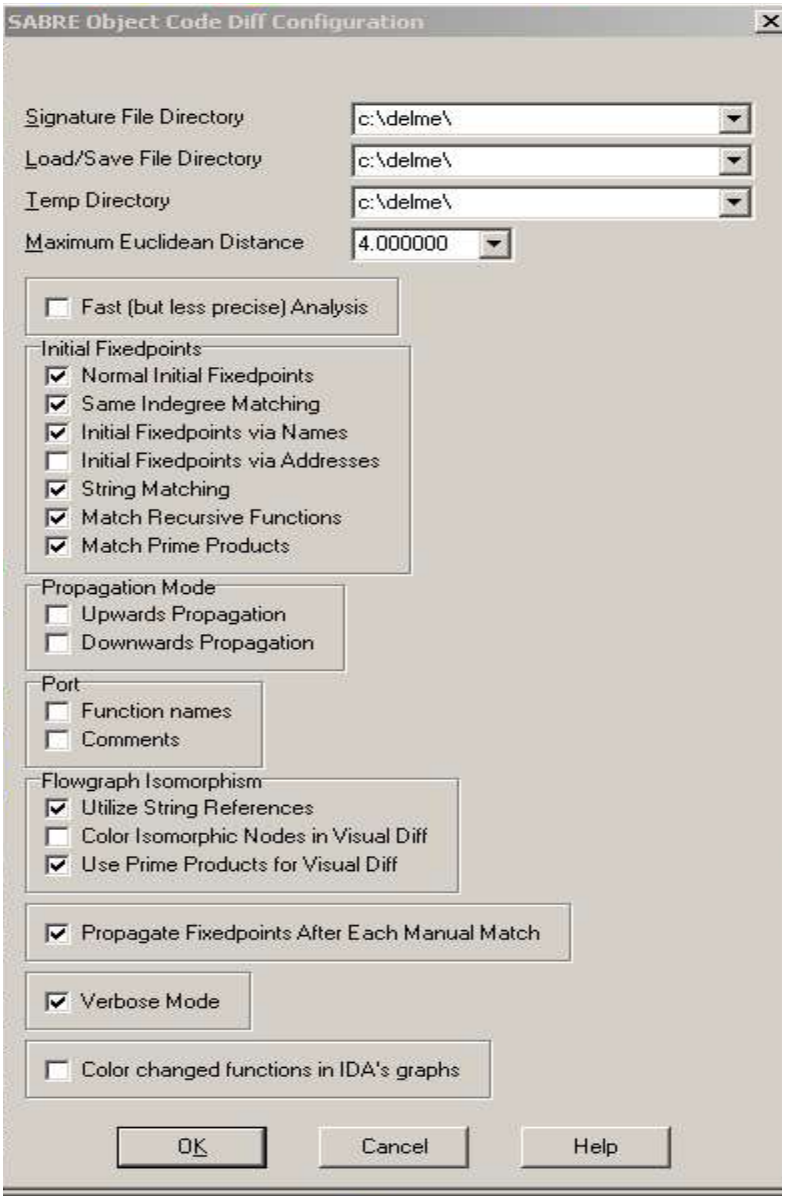

Illustration 2 The SABRE BinDiff v1.6 Configuration Screen

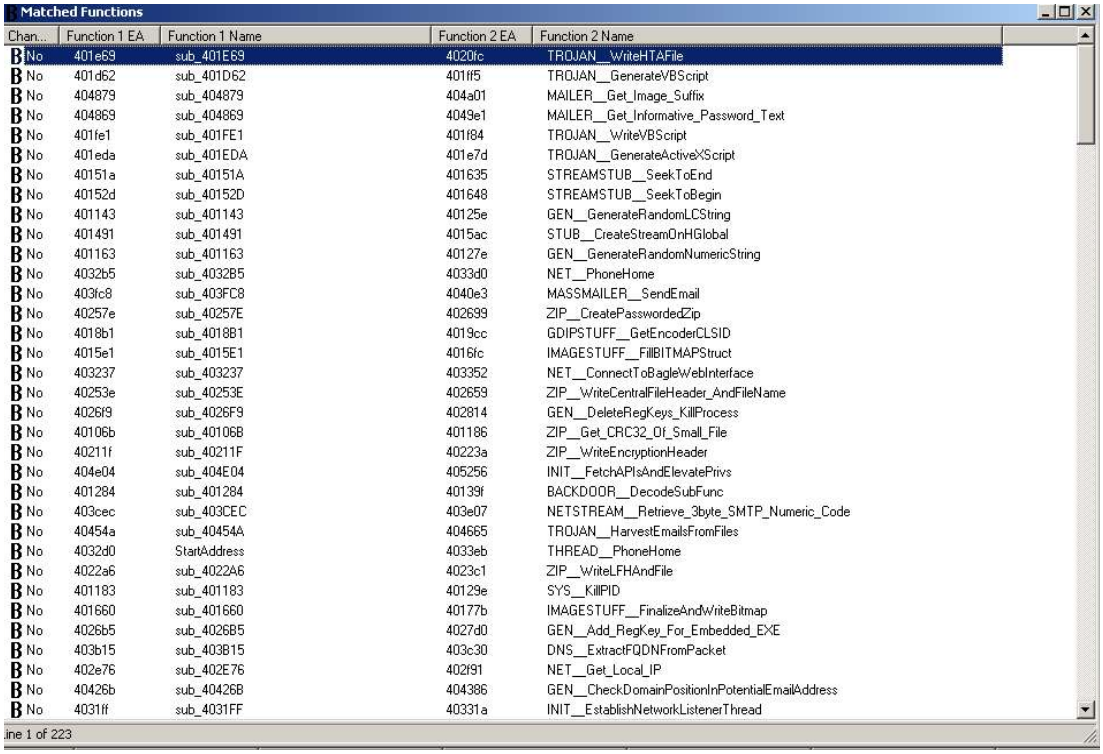

Illustration 3 The "Matched Functions" Window

A right-click on any function will pop up a context-sensitive menu. In order to port the comments and names from the Bagle/W disassembly, the first item on that menu ("Port") has to be selected. A warning will pop up asking whether we truly want to overwrite the names in the current disassembly with those from the Bagle/W database, and after answering "Yes", the program will spend some time processing and porting the comments from the other disassembly. After the processing is done, we will have almost all the comments and names from the Bagle/W disassembly already applied in the Bagle/X disassembly:

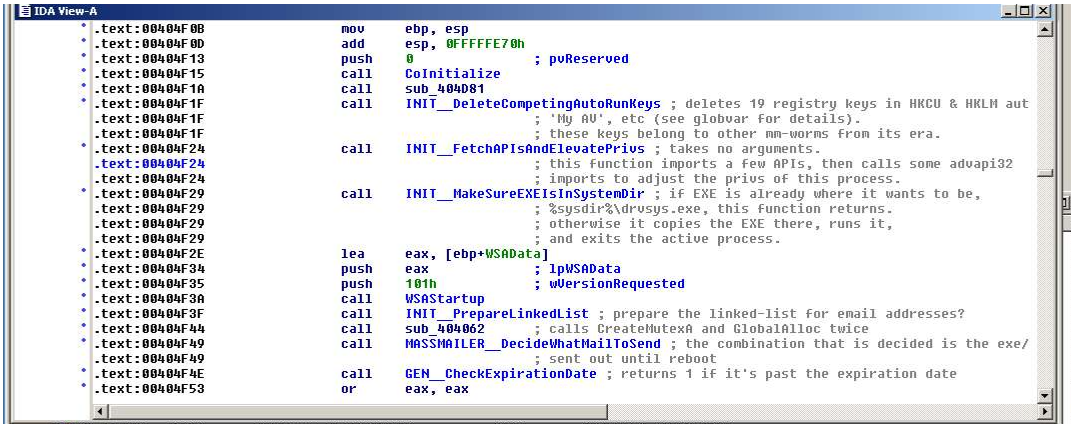

Illustration 4 The database after names and comments were ported in

The entire process took less than 5 minutes. All there is left to do now is examining and documenting the remaining 12 functions in the current disassembly that were not automatically mapped.