A New Bioinformatics-Inspired and Binary Analysis: Coding Style/Motif Identification

Scott Miller

Offensive Computing

For that B guy

Summary

After drawing an analog from computer binary analysis to a similar analysis problem in Biology, a useful technique modeled after BLAST was developed. This technique has been implemented with the binBLAST tool suite, a collection of command line tools with an optional web interface. This tool suite allows consideration of binary executables on a per-instruction level that does not require executables to follow any coding or interface standards. Of particular interest, this technique seems sensitive to coding and interface standards and creates the potential for automatic code/style identification.

Acknowledgements

- NSF/Scholarship for Service
- NMIMT/CS Department
- Thesis committee
- Technical and general reviewers
- Offensive Computing

Overview

- Concept
- binBLAST program suite
- Application: Version analysis
- Work-in-progress

Motivation

System Defense

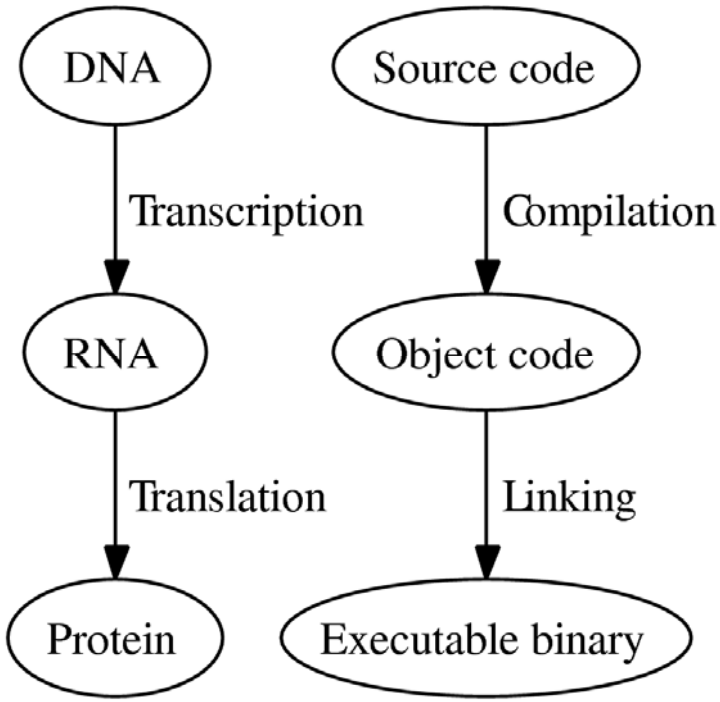
Current defense: policy, software, hardware

- New threats necessarily require patches
- Patches require verification
- Time available for verification is decreasing

The very tools we use to defend ourselves are contributing to the problem.

Bioinformatics Analogy

Functional analysis from code is not new…



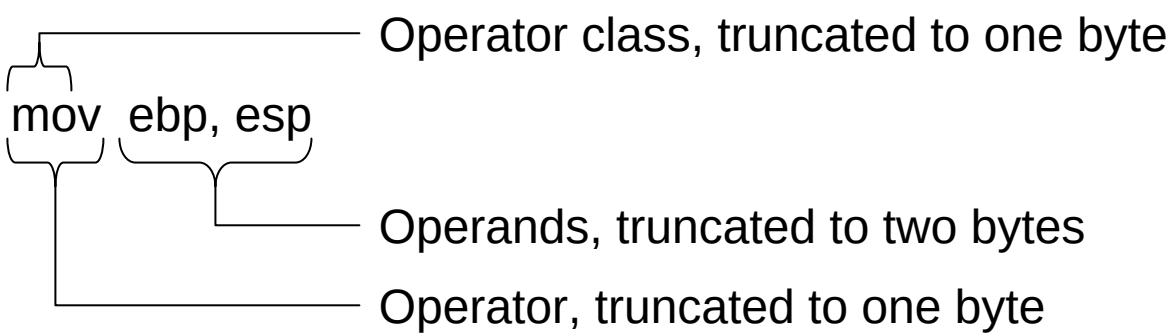
Technique
Alignment Scoring Example

- Same concept as BLAST
- Scored string matching
  - Match adds one (+1)
  - Difference subtracts one (-1)
  - Actual scoring constants determined using Karlin-Altschul analysis

Align #1		T	H	E	S	E		
Align #2		T	H	E	S	I	S	
Score	0	1	2	3	4	3	2	3

Technique
Scoring for Binaries

- Modified BLAST approach
- Karlin-Altschul analysis on over 553,901,654 instructions
- Instructions reduced to 4-byte format



Overview

- Concept
- **binBLAST program suite**
- Application: Version analysis
- Work-in-progress

binBLAST Suite

- Major analysis tools
  - mklib.py – builds disassembly library/index
  - bincompare – comparison engine
  - filterbincompare.py – output filter to clean/to refactor
  - matchoutput.py – visualization tools
- Auxiliary tools
  - binblast_html.cgi – web-based interface
  - objdumputil.py – debugger interface

binBLAST Suite

mklib.py

```
$ ./mklib.py --help
usage: mklib.py [options]

options:
  -h, --help            show this help message and exit
  -i FILE, --iso=FILE   Use an ISO file as the distribution
   source
  -d DIR, --dir=DIR     Use a directory tree as the
   distribution source
  -n MY_LINUX_X.X, --name=MY_LINUX_X.X
                        A unique identifier for this
   distribution
  -b, --bin             Produce binary output, name.dat and
   name.idx
  -v, --verbose         Produce verbose status information
  -o, --no-db           Do not produce the default database
   output, name.db
```

binBLAST suite

bincompare

```
$ ./bincompare --help
Usage: ./bincompare [options] fileA [offset [len]] fileB [offset
   [len]]
Options:
  -h Print this help
  -t=N: Set the threshold parameter to N, default is 10
  -x1=N: Set scoring parameter x1 to N, default is 6
  -x2=N: Set scoring parameter x2 to N, default is 5
  -x3=N: Set scoring parameter x3 to N, default is 4
  -x4=N: Set scoring parameter x4 to N, default is -4
Files:
  name - The name of the comparison file, i.e. name.dat
  offset - The start offset in quadwords
           Relative to the start of the file
           Defaults to 0
  len - The length for comparison in quadwords
           Defaults to the name's byte length / 4
 The fileA length must be smaller (or the same size)
  as fileB length.
```

binBLAST

filterbincompare.py

```
$ ./filterbincompare.py  --help
usage: filterbincompare.py [options]

options:
  -h, --help            show this help message and exit
  -d, --no-dirout       Do not correct dirout relative
  offsets to absolute offsets
  -n, --nosort          Don't sort results (reduces
  memory usage)
  -t MINSCORE, --threshold=MINSCORE
                        Threshold minimum score, if
  score is above Karlin-Altschul threshold
  -l MINLEN, --minlen=MINLEN
                        Minimum match length
  -a, --filea           Reduce results into matches
  covering only fileA
```

binBLAST

matchoutput.py

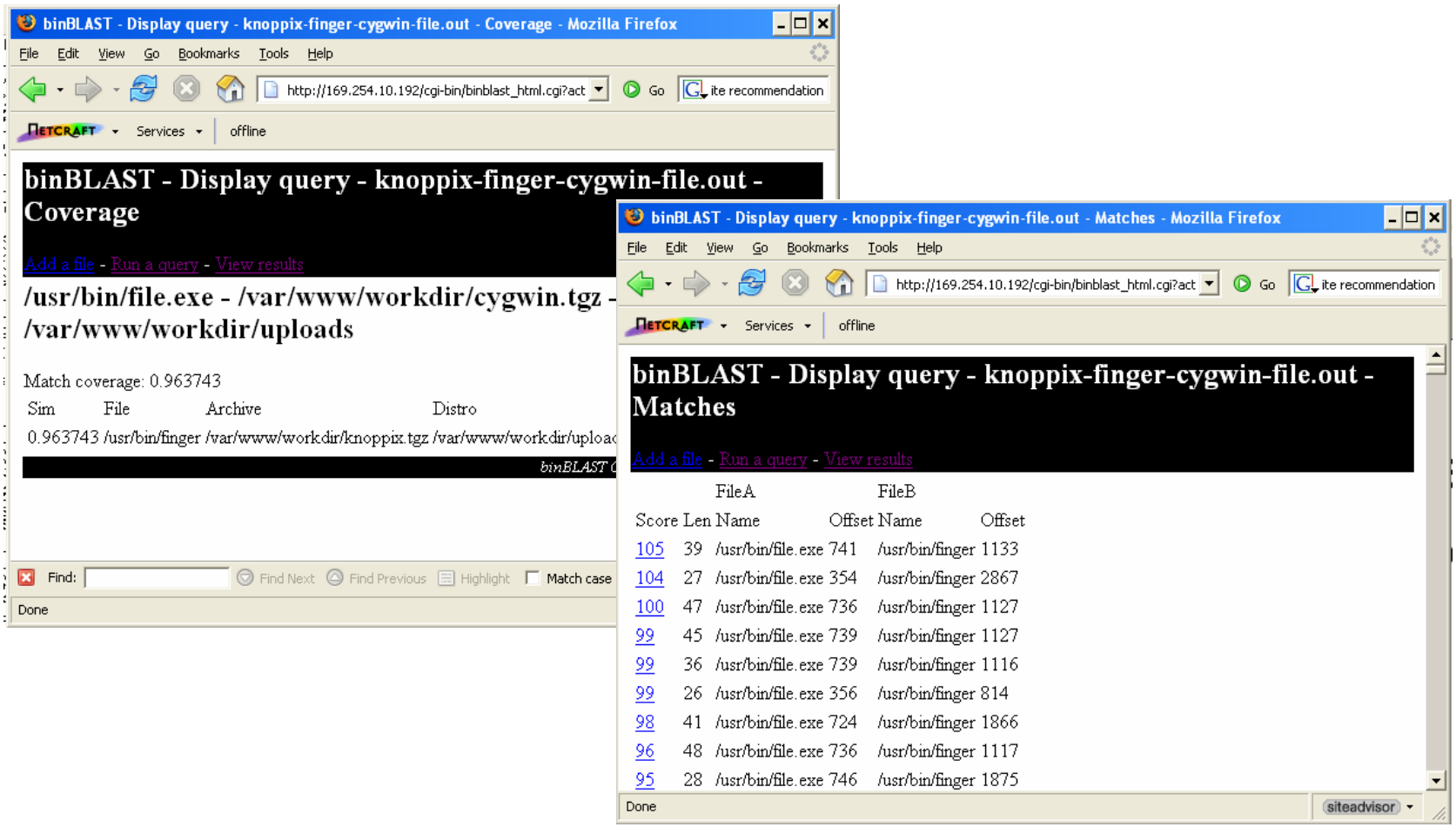
```
$ ./matchoutput.py --help
usage: matchoutput.py [options]

options:
  -h, --help          show this help message and exit
  -s, --similarity    Create a similarity table (LaTeX
  format)
  -d, --distance      Create a distance table (PHYLIP
  format)
  -m, --motifs        Create motif groups in graph
  -g, --graph         Create a graph of the matches
  (graphviz dot format)
  -a, --assembly      Print aligned assembly (requires
  IDXFILEA STARTA,LENA IDXFILEB STARTB,LENB as
  arguments)
```

binBLAST Suite
Process Overview/Demo

- Add programs using mklib.py
- Perform comparison using bincompare
- Pipe output of bincompare through filterbincompare.py
- Pipe output of filterbincompare.py through matchoutput.py

Except for mklib.py, tools use stdin/stdout for data passing

binBLAST
Web Interface



Overview

- Concept
- binBLAST program suite
- **Application: Version analysis**
- Work-in-progress

Application: Version Analysis Sendmail

Program	Distribution	Date released	Likely compiler
postfix-2.2.8	SUSE (i386)	1/7/2006	gcc 3.4.5
chromium-0.9.12	SUSE (i386)	9/13/2005	gcc 3.4.4
sendmail-8.13.5	SUSE (i386)	9/16/2005	gcc 3.4.4
sendmail-8.13.4	Fedora (i386)	3/27/2005	gcc 3.3.6
sendmail-8.12.9	Mandrake (i586)	3/29/2003	gcc 3.2.2
sendmail-8.9.1	Redhat (i386)	7/7/1998	gcc 2.8.1

Application: Version Analysis Raw Results

Score	Len	File A			File B		
		Name	Section	Off	Name	Section	Off
5095	899	sendmail-8.13.5	_init	0	sendmail-8.13.4	_init	0
4829	905	sendmail-8.12.9	_init	0	sendmail-8.13.4	_init	0
4797	899	sendmail-8.12.9	_init	0	sendmail-8.13.5	_init	0
4588	931	sendmail-8.13.5	_init	5	sendmail-8.13.4	(_init)	11
4488	907	sendmail-8.12.9	(_init)	32	sendmail-8.13.4	_init	5
...	...	...	...	...	...	...	...

Top results are associated with GCC-3 automatic code templates

Code templates common in GCC-3

Typical GCC-3 entry point

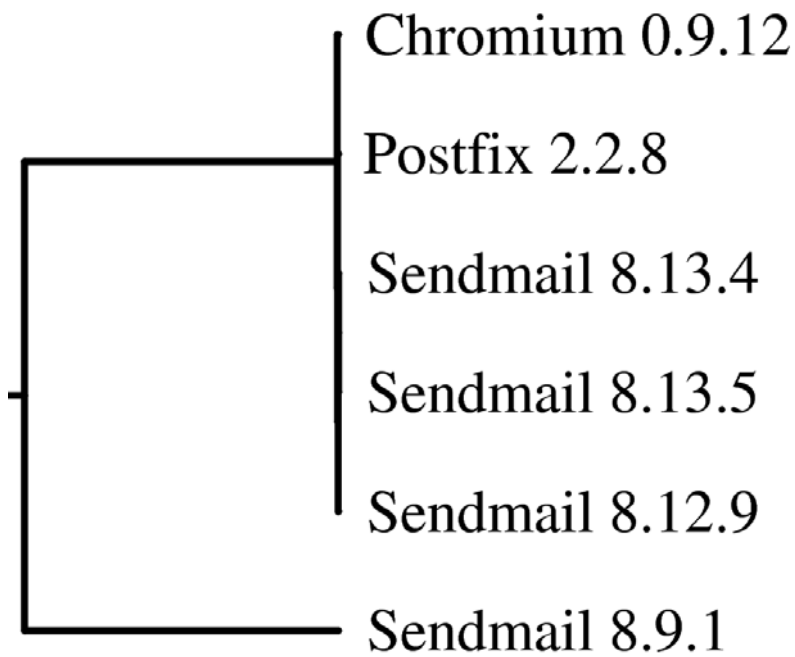
GCC-3 external symbol resolutions

Most common GCC-3 external call invocation

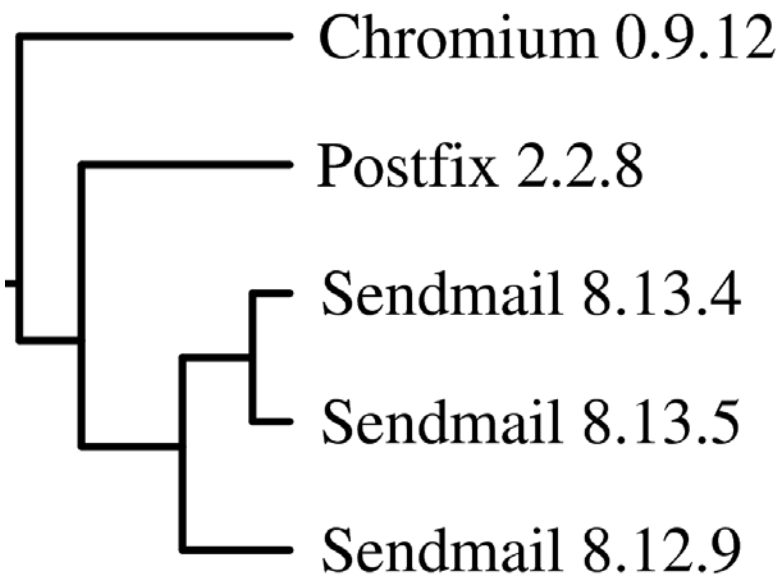
sendmail-8.13.5-1.i386.rpm		sendmail-8.13.4-2.i386.rpm	
_init:		*_init:*	
push	ebp	push	ebp
mov	ebp,esp	mov	ebp,esp
sub	esp,0x8	sub	esp,0x8
call	9f08 (chroot@plt+0x4c)	call	9f08 (__memmove_chk@plt+0x48)
call	9f88 (chroot@plt+0xcc)	call	9f88 (__memmove_chk@plt+0xc8)
call	991bc (sleep+0x2b82)	call	9a048 (sleep+0x2b96)
leave		leave	
ret		ret	
SSL_CTX_set_tmp_ rsa_calback@plt-0x10:		*SSL_CTX_set_tmp_ rsa_callback@plt-0x10:*	
push	DWORD PTR [ebx+4]	push	DWORD PTR [ebx+4]
jmp	DWORD PTR [ebx+8]	jmp	DWORD PTR [ebx+8]
add	BYTE PTR [eax],al	add	BYTE PTR [eax],al
SSL_CTX_set_tmp_ rsa_calback@plt:		*SSL_CTX_set_tmp_ rsa_callback@plt:*	
jmp	DWORD PTR [ebx+12]	jmp	DWORD PTR [ebx+12]
push	0x0	push	0x0
jmp	8c3c (_init+0x18)	jmp	8c20 (_init+0x18)
...		…	

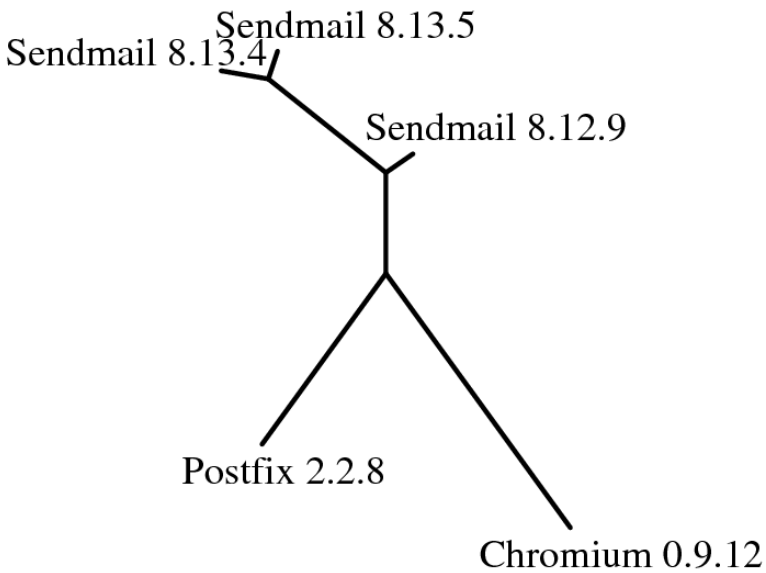
Application: Version Analysis
Phylogenetic Analysis

- PHYLIP Unweighted Pair Group Method with Arithmetic mean (UPGMA)
- Sendmail 8.9.1 distant because of use of GCC-2 compiler instead of GCC-3



Application: Version Analysis
Phylogenetic Analysis





Both the UPGMA (left) and Nearest-Neighbor (right) methods generate similar phylograms that reflect known relationships.

Review

- BLAST technique can be used on binaries
- binBLAST: an open-source analysis suite
- Sensitive to compiler code generation
- Detected relationships that mirror known relationships

Overview

- Concept
- binBLAST program suite
- Application: Version analysis
- **Work-in-progress**

Work in Progress

- Improved web interface
- Automatic malware classification
- Automatic antivirus signature generation
- Motif identification/sub-alignment match removal
- Wiki-type collaborative RE environment

Suite available at
http://www.offensivecomputing.net

Selected References

- BLAST: S. Altschul et al., *Basic Local Alignment Search Tool*, 1990
- Malware genomics: E. Carrera et al., *Digital Genome Mapping - Advanced Binary Malware Analysis*, 2004
- Malware: *Offensive Computing*, 2006
- Alias analysis: S. Debray et al., *Alias Analysis of Executable Code*, 1998
- Structural analysis: H. Flake, *Structural Comparison of Executable Objects*, 2004
- PHYLIP: J. Felsenstein, *PHYLIP*, 2005
- `Packers': M. Oberhumer et al., *The Ultimate Packer for eXecutables*, 2004.