Recovering the Toolchain Provenance of Binary Code

Nathan Rosenblum
July 19, 2011

nater@cs.wisc.edu
http://pages.cs.wisc.edu/~nater/
Joint work with

Barton Miller and Xiaojin Zhu
Program Provenance
Why Provenance?
Why Provenance?
Why Provenance?
Why Provenance?
Why Provenance?
Why Provenance?
Why Provenance?

“user compiled Firefox with buggy GCC”
Provenance Recovery
Provenance Recovery

Statistical Machine Learning
program binary
program binary

... 55 89 e5 83 ec 2c 57 56 53 8b 45 0c 8b 00 a3 90 a3 05 08 85 c0 74 2b 83 c4 ...
... 55 89 e5 83 ec 2c 57 56 53 8b 45 0c 8b 00 a3 90 a3 05 08 85 c0 74 2b 83 c4 ...

<push EBP ; * ; mov ESP, EBP>   <mov [IMM], RAX ; sub [IMM], RAX>
program binary

... 55 89 e5 83 ec 2c 57 56 53 8b 45 0c 8b 00 a3 90 a3 05 08 85 c0 74 2b 83 c4 ...

\texttt{\textless{}push EBP ; * ; mov ESP, EBP\textgreater{} \textless{}mov [IMM], RAX ; sub [IMM], RAX\textgreater{}}

Control Flow Graph
... 55 89 e5 83 ec 2c 57 56 53 8b 45 0c 8b 00 a3 90 a3 05 08 85 c0 74 2b 83 c4 ...

<push EBP ; * ; mov ESP, EBP>  <mov [IMM], RAX ; sub [IMM], RAX>

Control Flow Graph

layout, block contents
... 55 89 e5 83 ec 2c 57 56 53 8b 45 0c 8b 00 a3 90 a3 05 08 85 c0 74 2b 83 c4 ...

\[\text{mov [IMM], RAX \; sub [IMM], RAX}\]

Control Flow Graph

Call Graph
program binary

... 55 89 e5 83 ec 2c 57 56 53 8b 45 0c 8b 00 a3 90 a3 05 08 85 c0 74 2b 83 c4 ...

<push EBP ; * ; mov ESP, EBP>  <mov [IMM], RAX ; sub [IMM], RAX>

Control Flow Graph
Call Graph
External Libraries

layout, block contents

fprintf
Instruction-level Features
IDIOMS

```c
push EBP ; * ; mov ESP, EBP

mov [IMM], RAX ; sub [IMM], RAX
```
IDIOMS

single-instruction wildcard

<push EBP ; * ; mov ESP, EBP>

<mov [IMM], RAX ; sub [IMM], RAX>

opcode class abstraction

hidden immediates
IDIOMS

single-instruction wildcard

<push EBP ; * ; mov ESP, EBP>

<mov [IMM], RAX ; sub [IMM], RAX>

hidden immediates

opcode class abstraction

N-GRAMS

<4889c2be>  <018b45f8>  <8d45f8>

4-grams

3-grams
Control flow graph
Control flow graphlets
code element nodes
(e.g. basic blocks)
typed edges
(branch, call, etc.)
code element nodes (e.g. basic blocks)

node colors

typed edges (branch, call, etc.)
code element nodes
(e.g. basic blocks)

typed edges
(branch, call, etc.)

node colors

Ex: instruction summary graphlets

Color bit field

15 instruction categories

2^{15} possible colors

sparse in practice
code

feature vector
“decompiles to <push ebp, ...”

27 x
“decompiles to <push ebp, ...”
code

`byte string`

`function`

`whole program`

```
“decompiles to push ebp, ...”
```

`feature vector`

27 x
Evaluation & Results
Evaluation

175 programs $\times$ 2,686 binaries $\rightarrow$ 955k functions
Evaluation

175 programs ×

Language

Acc. .999

2,686 binaries → 955k functions
Evaluation

175 programs \(\times\) 2,686 binaries \(\rightarrow\) 955k functions

Language

Compiler

Acc. .999 .998
Evaluation

175 programs \times 2,686 binaries \rightarrow 955k functions

Language

Compiler

Optimization

Acc. \cdot 0.999 \cdot 0.998 \cdot 0.993
Evaluation

175 programs × 2,686 binaries → 955k functions

Language

Compiler

Optimization

Version

Acc.  .999  .998  .993  .910

Tuesday, July 19, 2011
correct

wrong

opt. level

statically linked library code
correct

statically linked library code

wrong opt. level

wrong version
correct

statically linked library code

wrong opt. level

wrong version

MSVS 2003 ↔ MSVS 2005 ↔ MSVS 2008
Summary

Program + Learning Framework

01110101
10101010
10110100
10110101

Tuesday, July 19, 2011
questions