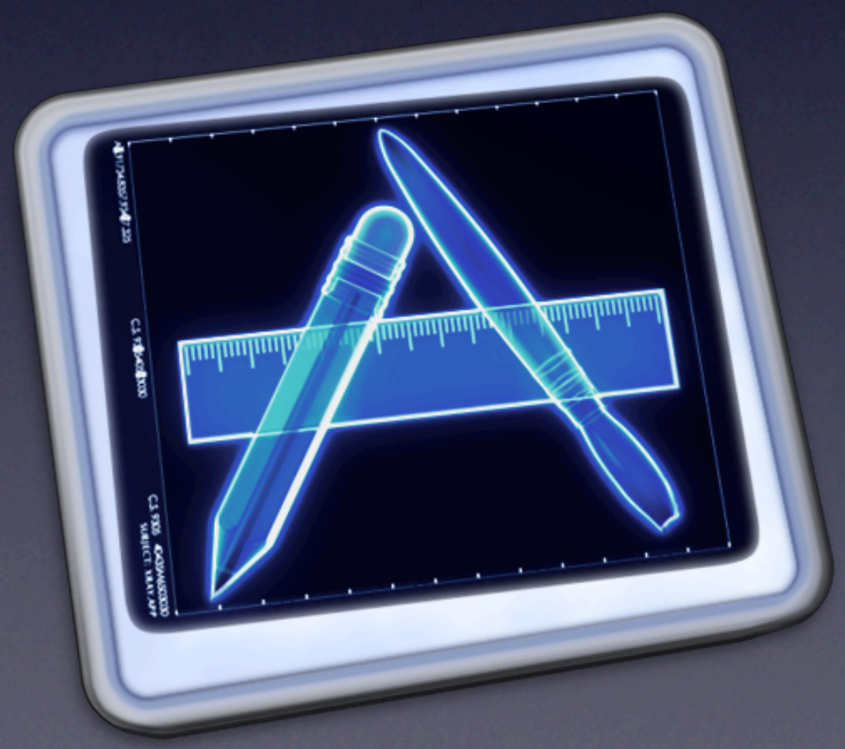
Obfuscation of Executable Code

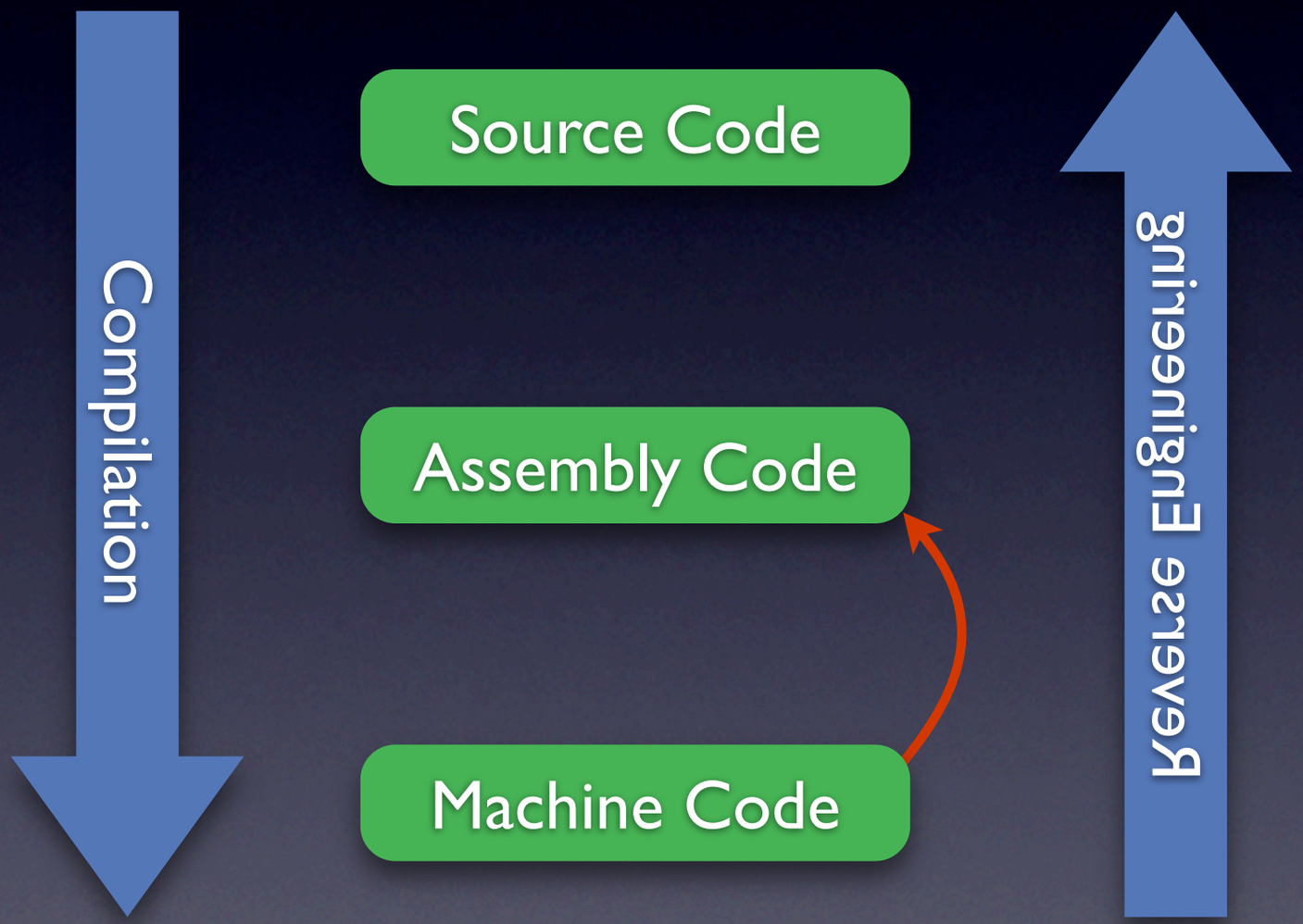
Cullen Linn, Saumya Debray



Motivation

- DRM is (still) a reality due to (fading?) industry demand
- DRM systems are successful (see iTunes)
- even though they have been hacked
- 100% solutions seem overkill
- obfuscation gets you 95% of the way with considerably less complexity and effort

Reverse Engineering



Disassembling

- problem: finding instruction boundaries
- linear sweep
  - can mistake data for code
- recursive traversal
  - indirect jump heuristic

Confusion

- goal: confuse the disassembler's notion of instruction boundaries
- solution: insert junk bytes
- but: IA-32 disassembly self-synchronizes
- solution: choose junk that prolongs resync the farthest

Junk Insertion

- junk must not be reachable at runtime
- insert junk after basic blocks that end with an unconditional jump
- thwarts linear sweep disassemblers
- 15% confusion

More Junk

- branch flipping:

$b_{\neg cc}$ L’

b_{cc} Addr ⟶ jmp Addr

L’:

- 37% confusion
- call conversion: return with offset
- 42% confusion

Confuse Recursion

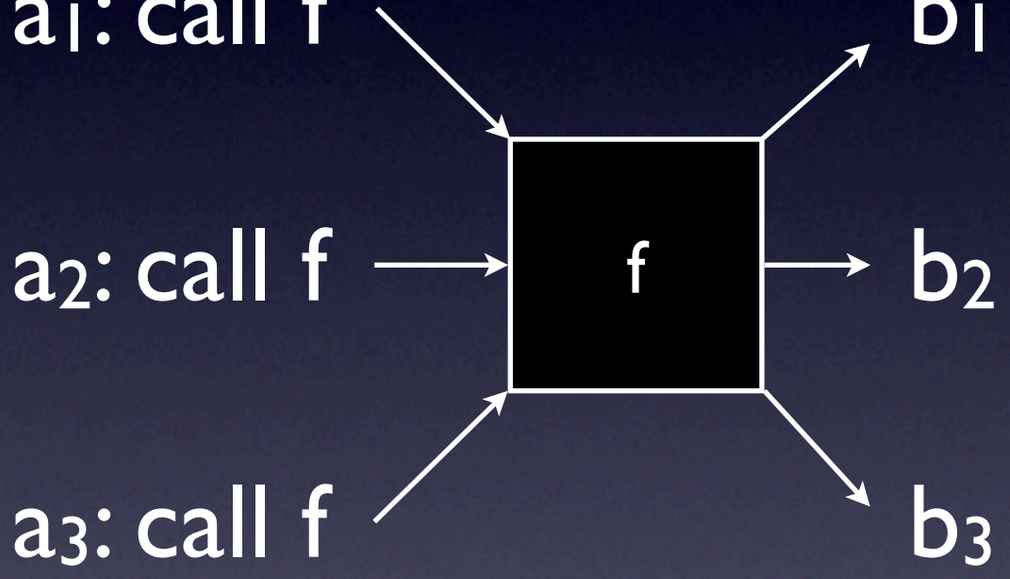
- thwart recursive traversal
- obfuscate control transfer targets
- bogus control transfer targets

Branch Functions

a_1: jmp b_1 ⟶ b_1

a_2: jmp b_2 ⟶ b_2

a_3: jmp b_3 ⟶ b_3



Bogus Targets

- call conversion
- opaque predicates
- jump tables with bogus entries
- insert junk at the bogus target
- the last two are not implemented

Evaluation

- confusion factor: fraction of incorrect instructions / basic blocks / functions
- IDA Pro
  - 66% instructions
  - 81% basic blocks
  - 85% functions

Evaluation

- threshold to trade slowdown against confusion
- maximum slowdown observed: 5x
- code size increase negligible

Discussion

- Is this useful for anything other than DRM?
- Who will win the arms race?
- Other examples for 90% solutions with 10% of the effort?
- „DRMs haven’t worked, and may never work, to halt music piracy.“ (Steve Jobs)