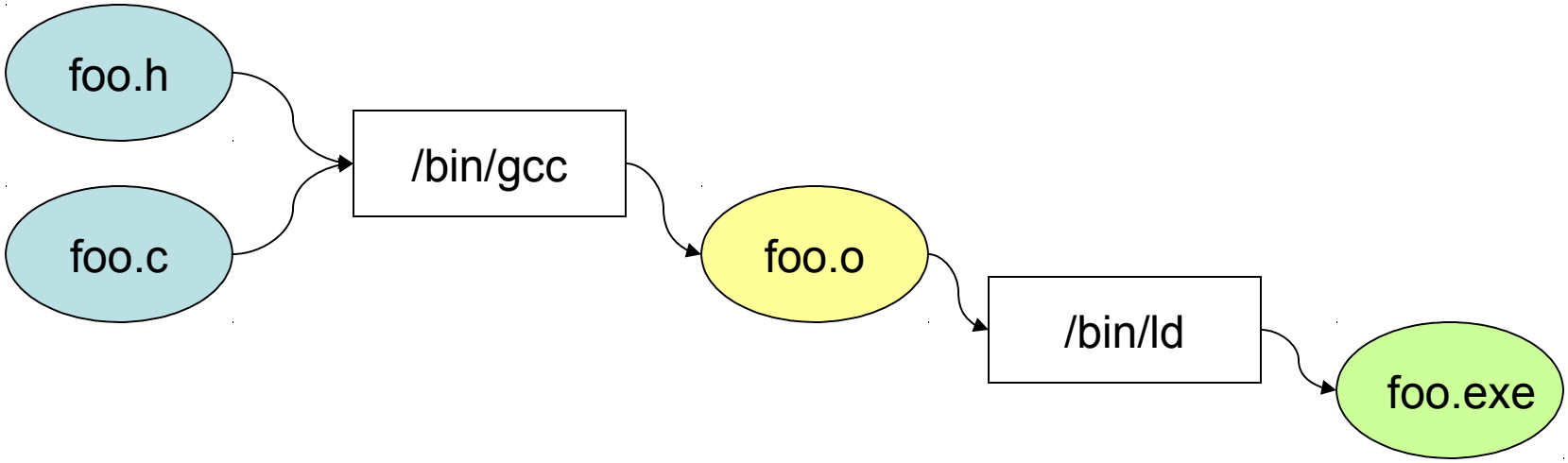
Back to sources: what's in your binary?

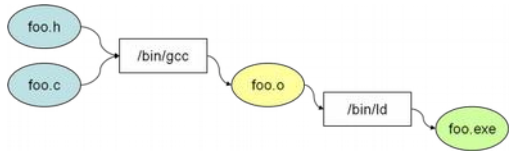
FOSDEM 2017: Tracing back from a binary to its corresponding source code



nexB

Corresponding Source?

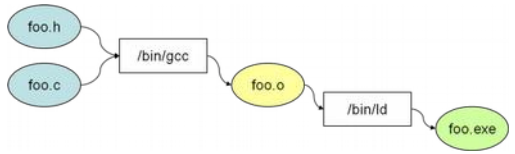
- You build and distribute binaries, right?
- What is the corresponding source code?
- Why?
  - license compliance and/or enforcement
  - code hygiene and build sanity



Techniques to get there

- Forward tracing
  - Dynamic build tracing
- Backward tracing
  - Dependencies analysis
  - Symbols and debug symbols
  - Shared string content
  - Build log analysis
  - Compiler conventions (e.g. Java)

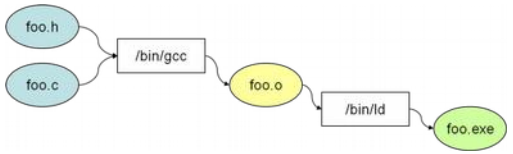
nexB



The problems

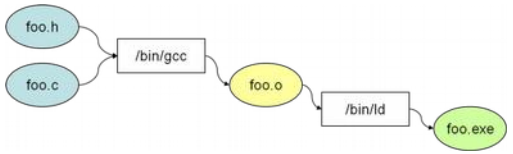
- Static analysis is not an exact science
- Difficult to obtain what is needed from dev teams
- Debug builds are not the norm
- Builds are complex and hard to modify
- Hard to conclude that something is NOT built
- Built != Deployed

nexB



The ideal solution

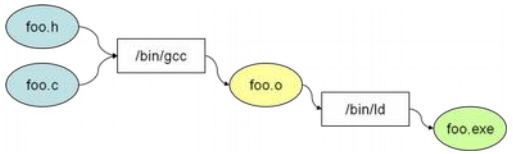
- Should be very easy on the dev team
- NO CHANGE to the build and config
- Should provide 20/20 vision in analysis
- Should be 100% accurate



Syscalls to the rescue!

- The “machine” language of the Kernel
- 100% accurate
- Everything that touches a file, access the network ends up in a syscall
- Very low level e.g.:
  - open/read/write file
  - spawn process and run executable

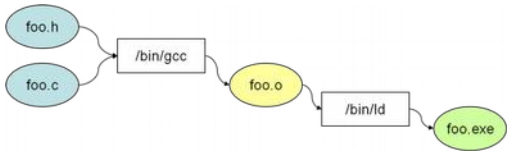
nexB



TraceCode

- Run a build under “strace”
- Collect a trace of EVERY syscalls
- Parse the trace
- Rebuild a graph of the file transformation that took place during the build
- Query the graph for fun and profit!

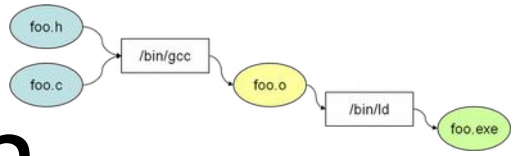
nexB



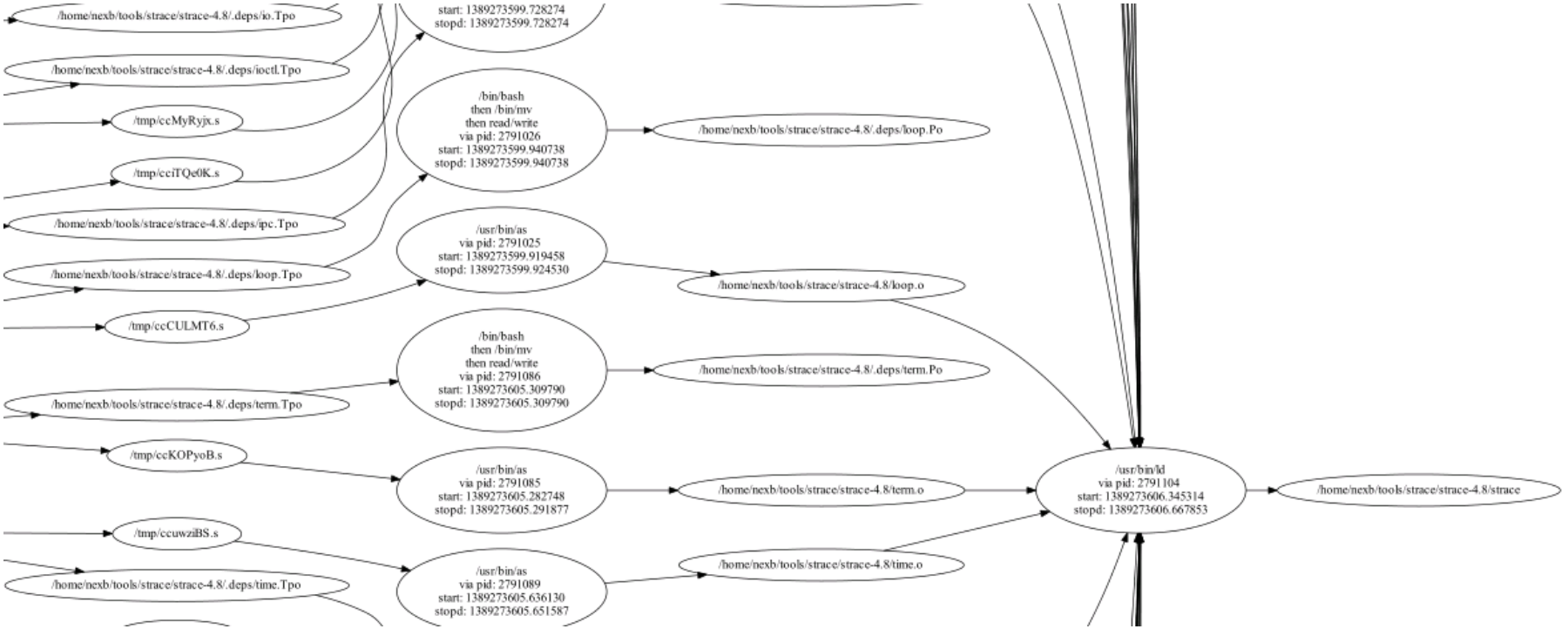
TraceCode (2)

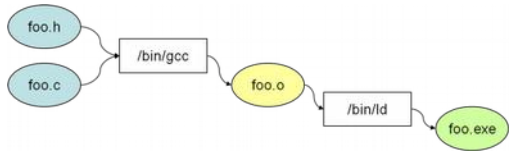
- Completely agnostic wrt. toolchain or programming language
- Does not require *ANY* change to the build process
- Only need to run a traced build
- Can provide 20/20 vision in the build process

nexB



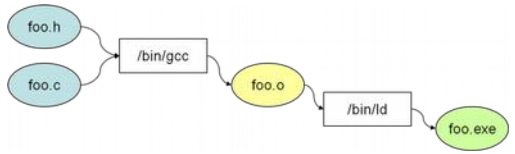
TraceCode graph





Step by step

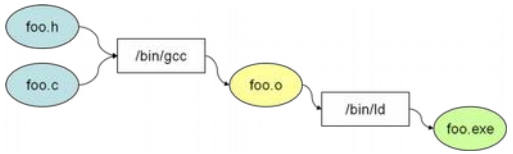
- Install strace
- Run the build run under strace
- Install and run TracCode on the trace



Other techniques

- Debug symbols: DWARFs and similar
- Symbols: ELFs and similar
- Build logs: brittle, messy
- Shared string content: quite promising!
  - Collect sources and builds
  - Find the unique strings present in both
  - Index these strings (e.g. hashes or aho-corasick)
  - Scan an arbitrary binary for matching strings

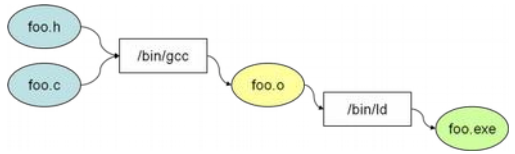
nexB



Credits

- strace rocks!
- TraceCode is implementing this paper "Discovering Software License Constraints: Identifying a Binary's Sources by Tracing Build Processes" http://www.st.ewi.tudelft.nl/~sander/pdf/publications/TUD-SERG-2012-010.pdf
- By Sander van der Burg, Julius Davies, Eelco Dolstra, Daniel M. German, Armijn Hemel.

nexB

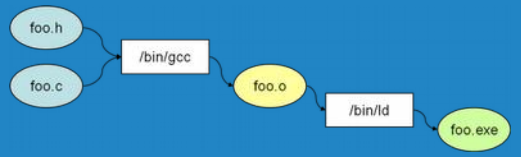


Get TraceCode

- https://github.com/nexB/tracecode-build/
- Written in Python, Apache-licensed.

- Other tools “about code”: http://aboutcode.org
  - **ScanCode**: Scan your code for license, copyrights, packages, etc.
  - **AboutCode manager**: GUI to Review scans. Document origin and license conclusions
  - **AttributeCode**: Document the origin and license of thirdparty code. Collect inventories, generate attribution notices

nexB



Thank you!

Questions?

Credits

Special thanks to all the people who made and released these awesome free resources:

- Presentation template by SlidesCarnival
- Photographs by Unsplash
- And all the software authors that made TraceCode possible