Software Diversity
1 concept and 10 papers I Love

Benoit Baudry
Professor, KTH
Risks of monoculture
- No specialization
- Same bugs
- Same vulnerabilities
Software diversity mitigates the risks of software monoculture with diverse behaviors.
Software diversity

SRSLSLRSRLLSSRRLRL

Illustration inspired by G. Berry. « A la chasse aux bugs, la maladie du certain » (8 juin 2011)
Software diversification

SRSLSLRSRLLSSRRLRL
Software diversification

SRSLSLRSRLSSLSSRRLRL variant
Software diversification

SRSLSLRSRLLSSRRLRL variant
Software diversification

SRSLSLRSRLLSSRRLRL
SRSLSLSSRLLSSRRLRL
Software diversification exploits the extraordinary resources of runtimes, languages and randomness.
A journey into software diversity
Precursors

• Brian Randell. System structure for software fault tolerance. 1975.
Pioneers of automatic diversification

• Fred Cohen, 1993
  • Increase the costs of attacks
  • Program transformations
  • Pioneer: reordering, garbage insertion, function mix

• Stephanie Forrest, 1997
  • Biological inspiration
  • Avoid unnecessary consistency
  • Pioneer: NOP insertion, random memory padding
  • Prototype of randomized stack layout
Address space layout randomization

• PaX Linux kernel patch. 2000.
  • Separate readable data pages and executable code pages
  • Address space layout randomization: heap, stack and libraries

• ASLR is now in all main Oss
  • Mitigates ret-to-libc and stack smashing

https://en.wikipedia.org/wiki/PaX
Shacham, H. and colleagues. On the Effectiveness of Address-Space Randomization. 2004
NOP insertion

- Compiler-based diversification
  - Randomly insert NOPs in the generated binary
  - One different binary at each compilation
- Mitigates return oriented programming

Good enough software

- Functionality removal, computation discard
- Mitigate homogeneous performance

```
for (i = 0; i < n; i++) { ... }
```

```
for (i = 0; i < n; i += 2) { ... }
```
Mutational robustness

- Source can be randomly transformed with speculative transformations
- Empirical evidence of software mutational robustness
- Mitigates risks of bug and vulnerability monoculture

Moving Target Defenses

• Runtime evolution + diversity
Conclusion

• The forces of monoculture are strong
  • Technical standards (e.g., JSON)
  • Socio-technical networks (e.g., Github)
  • The penetration of software in society (e.g., Wordpress)

• Extraordinary challenges to fuel software diversity
  • Remodel the natural diversity of code strata
  • Embrace evolution with DevOps
  • Explore the space of short-lived data and programs