DevSecOps: Delivering Reliable and Secure Software Systems via Automated Bug Finding and Hardening

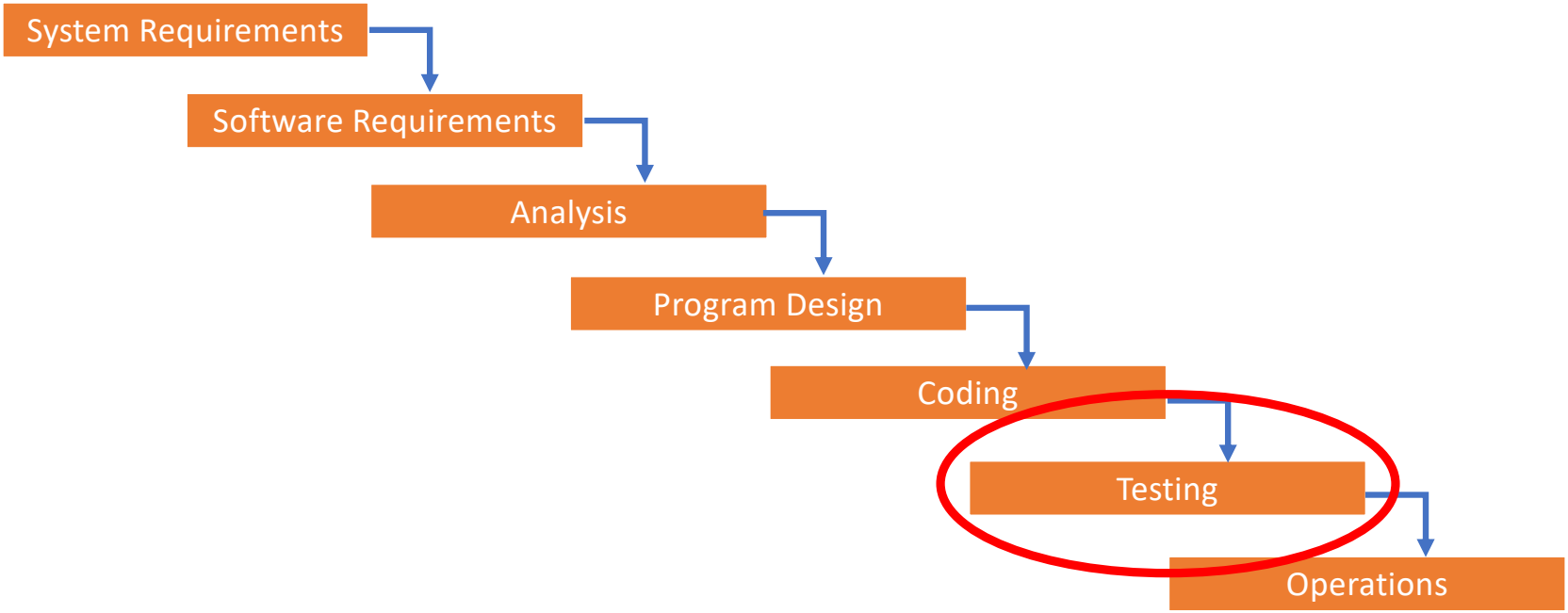
Jack Davidson
Department of Computer Science
University of Virginia
Contact: jwd@virginia.edu, http://sdg.cs.virginia.edu

SCHOOL of ENGINEERING & APPLIED SCIENCE

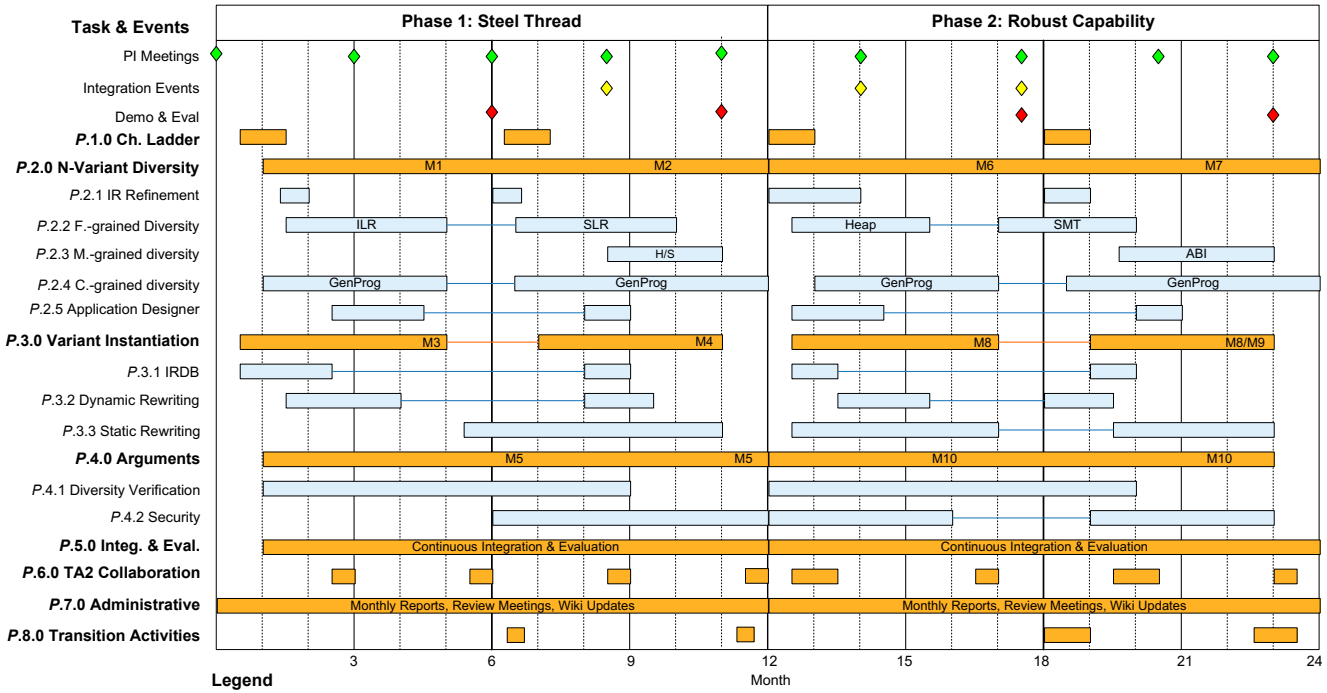
Outline of Talk

- Brief History of Software Development Methodologies
  - Waterfall
  - Agile
  - DevOps
- Left Shift from DevOps to DevSecOps
- Fuzzing
- DevSecOps with Helix++
- Research Challenges and Resources

Waterfall Model (Royce)



Waterfall Model



Legend

M1: N-Variant ILR Prototype, **M2**: N-Variant SLR Prototype, **M3**: Dynamic rewriter evaluated on Challenge Ladder, **M4**: Static rewriter evaluated on Challenge Ladder, **M5**: Example proofs and arguments, **M6**: N-Variant Heap Prototype, **M7**: N-Variant ABI Prototype, **M8**: Dynamic rewriter evaluated on Challenge Ladder, **M9**: Static rewriter evaluated on Challenge Ladder, **M10**: Prototype proofs and arguments

Waterfall Model

SOW Task		Duration (Months)	Monthly Cost	Fully-Burdened Labor Cost Include any Fee						$000s
				Senior	Mid	Junior	Total	SubC	Conslt	Total
1.1.0	Challenge Ladder	2	3	3	8	6	17	17	0	34
1.2.0	N-Variant Diversity	11	56	81	205	141	427	247	0	674
1.2.1	IR Refinement	1.5	4	5	13	9	27	15	0	42
1.2.2	Fine-grained Div.	7	28	41	102	70	213	124	0	337
1.2.3	Medium-Grained Div.	2.5	7	10	26	18	53	31	0	84
1.2.4	Coarse-grained Div.	9.5	14	20	51	35	107	62	0	168
1.2.5	Application Designer	3	4	5	13	9	27	15	0	42
1.3.0	Variant Instantiation	8.5	17	29	73	50	152	51	0	202
1.3.1	IRDB	3	3	5	12	8	25	8	0	34
1.3.2	Dynamic Rewriting	4	4	7	18	13	38	13	0	51
1.3.3	Static Rewriting	5.5	10	17	42	29	88	29	0	118
1.4.0	Arguments	11	35	2	6	4	13	408	0	421
1.4.1	Diversity Verification	8	18	1	3	2	6	204	0	211
1.4.2	Security	6	18	1	3	2	6	204	0	211
1.5.0	Integ. & Eval.	11	14	16	40	28	84	84	0	168
1.6.0	TA2 Collaboration	2	7	8	20	14	42	42	0	84
1.7.0	Administrative	12	6	6	16	11	34	34	0	67
1.8.0	Transition Activities	1	3	3	8	6	17	17	0	34
	Phase 1 Total	**12**	**140**	**149**	**377**	**259**	**785**	**900**	**0**	**1684**
2.1.0	Challenge Ladder	2	3	3	8	6	17	17	0	33
2.2.0	N-Variant Diversity	12	56	78	196	135	409	259	0	668
2.2.1	IR Refinement	3	3	5	12	8	26	16	0	42
2.2.2	Fine-grained Div.	6	14	19	49	34	102	65	0	167
2.2.3	Medium-Grained Div.	3.25	14	19	49	34	102	65	0	167
2.2.4	Coarse-grained Div.	9.5	21	29	74	51	153	97	0	250
2.2.5	Application Designer	3	3	5	12	8	26	16	0	42
2.3.0	Variant Instantiation	8.5	17	29	72	50	150	50	0	200
2.3.1	IRDB	2	3	5	12	8	25	8	0	33
2.3.2	Dynamic Rewriting	3.5	4	7	18	12	38	13	0	50
2.3.3	Static Rewriting	8	10	17	42	29	88	29	0	117
2.4.0	Arguments	11	35	2	6	4	13	405	0	417
2.4.1	Diversity Verification	8	17	1	3	2	6	202	0	209
2.4.2	Security	8	17	1	3	2	6	202	0	209
2.5.0	Integ. & Eval.	12	14	16	40	28	83	83	0	167
2.6.0	TA2 Collaboration	3	7	8	20	14	42	42	0	83
2.7.0	Administrative	12	6	6	16	11	33	34	0	67
2.8.0	Transition Activities	2	3	3	8	6	17	17	0	33
	Phase 2 Total	**12**	**139**	**145**	**366**	**252**	**763**	**907**	**0**	**1670**
3.1.0	Challenge Ladder	4	3	5	15	0	20	20	0	40
3.2.0	N-Variant Diversity	17	49	125	376	0	501	91	0	593
3.2.1	Diversity Selection	17	33	84	251	0	334	61	0	395
3.2.2	Diversity Transforms	14	16	42	125	0	167	30	0	198
3.3.0	Variant Instantation	15	25	70	211	0	282	15	0	296
3.3.1	IRDB	4	2	5	14	0	19	1	0	20
3.3.1	Dynamic Rewriting	7.5	10	28	84	0	113	6	0	119
3.3.2	Static Rewriting	11	13	38	113	0	150	8	0	158
3.4.0	Arguments	10	31	2	7	0	9	366	0	375
3.4.1	Diversity Verification	9	16	1	4	0	5	183	0	188
3.4.2	Security	10	16	1	4	0	5	183	0	188
3.5.0	Integ. & Eval.	18	31	66	197	0	263	113	0	375
3.6.0	TA2 Collaboration	5	8	17	52	0	69	30	0	99
3.7.0	Administrative	18	8	12	37	0	49	49	0	99
3.8.0	Transition Activities	13	8	12	37	0	49	49	0	99
	Phase 3 Total	**18**	**165**	**311**	**932**	**0**	**1243**	**733**	**0**	**1976**
	Project Total	**42**	**444**	**605**	**1675**	**511**	**2791**	**2540**	**0**	**5330**

Figure 9: Cost Summary Table.

Waterfall Model

- Linear sequencing
- Changing anything is hard
- DOD requires an Engineering Change Proposal (ECP)
  - Takes months to get approved

Agile History

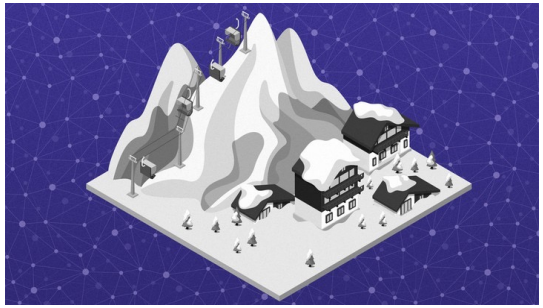
- Meeting at Snowbird, UT February 21, 2001
- Created statement of values
- See: “The Winter Getaway That Turned the Software World Upside Down. Caroline Mimbs Nyce, The Atlantic, December 8, 2017.

TECHNOLOGY

The Winter Getaway That Turned the Software World Upside Down

How a group of programming rebels started a global movement

CAROLINE MIMBS NYCE DECEMBER 8, 2017



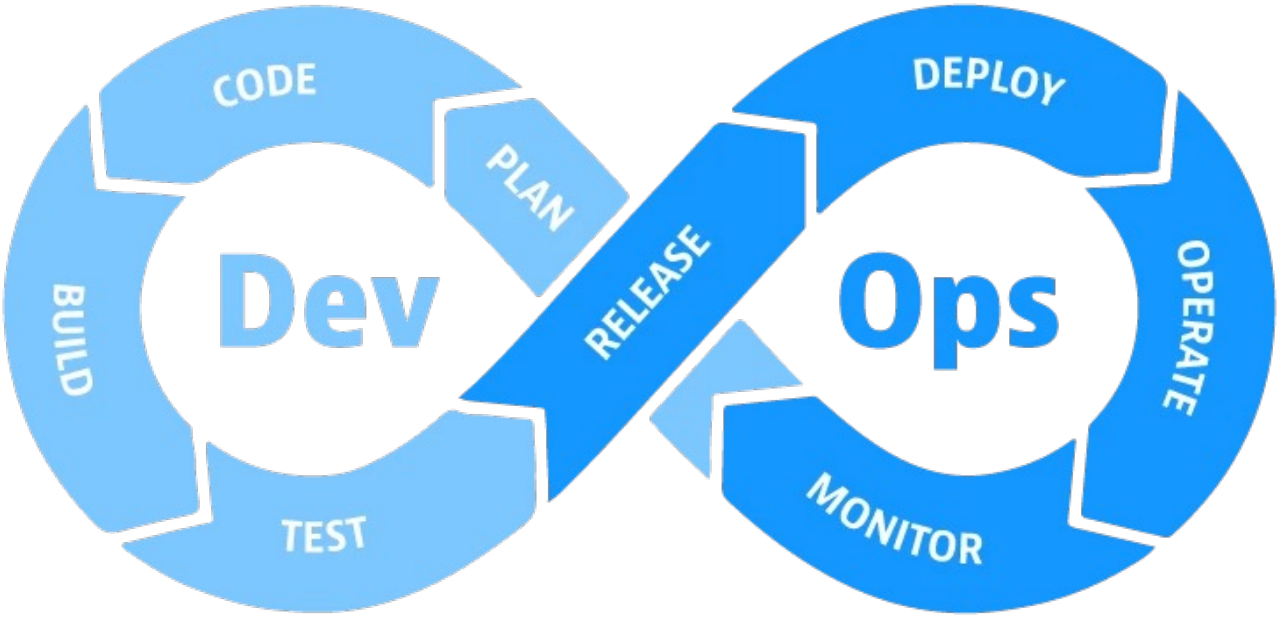
Manifesto for Agile Software Development: Four Values and Twelve Principles

Values

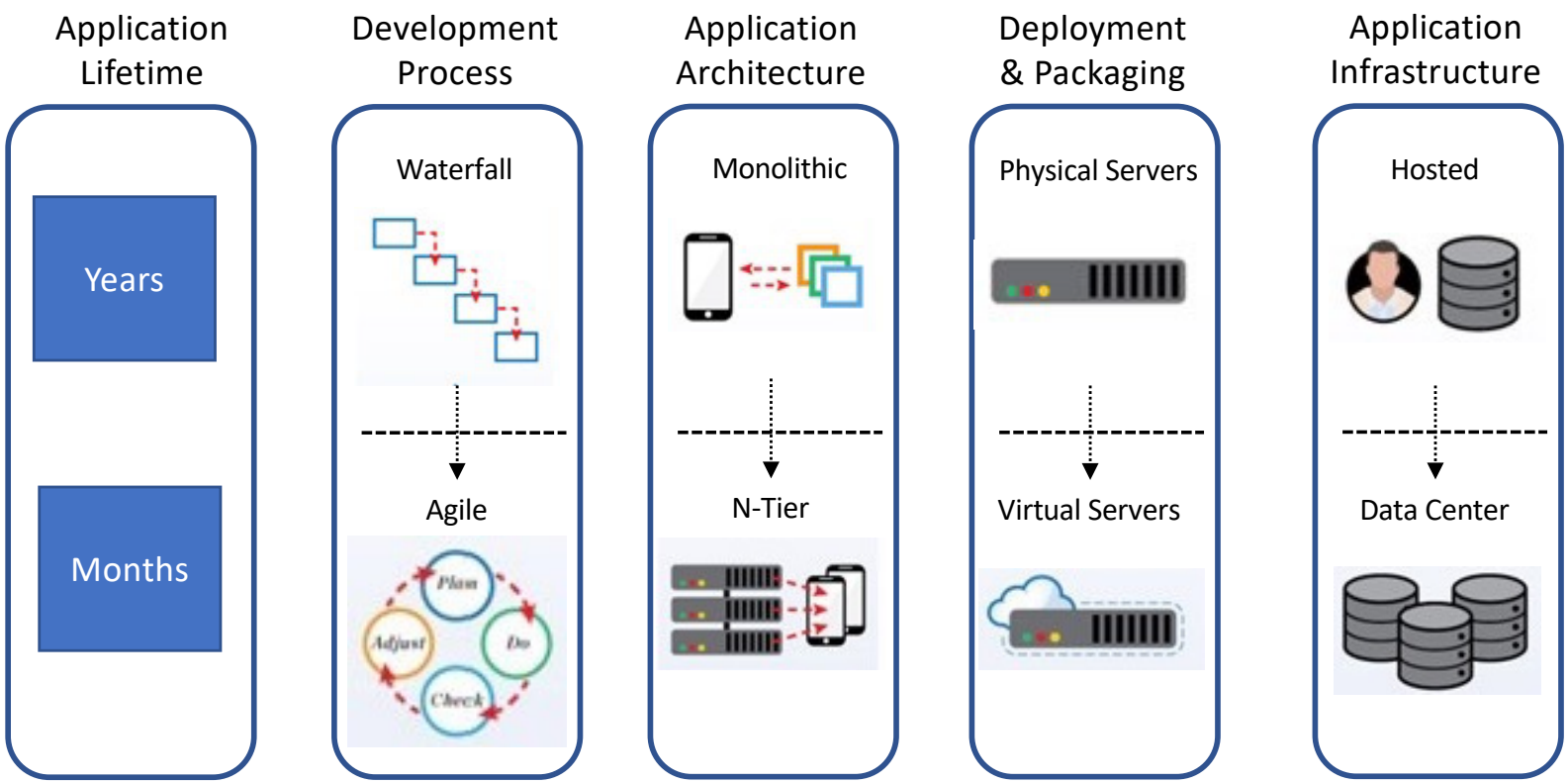
1. **Individuals and interactions** over processes and tools
2. **Working software** over comprehensive documentation
3. **Customer collaboration** over contract negotiation
4. **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

DevOps


CODE
PLAN
BUILD
Dev
TEST
RELEASE
DEPLOY
Ops
OPERATE
MONITOR

Waterfall to DevOps



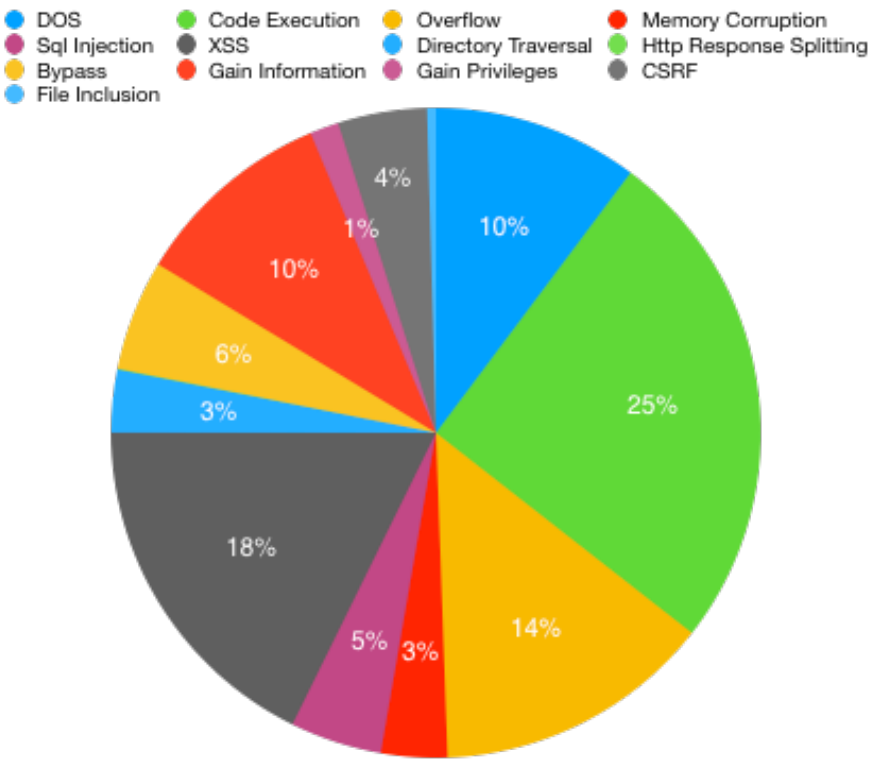
Source: Nicholas Chaillan, OSD

UNIVERSITY of VIRGINIA | SCHOOL of ENGINEERING & APPLIED SCIENCE

Outline of Talk

- Brief History of Software Development Methodologies
  - Waterfall
  - Agile
  - DevOps
- Left Shift from DevOps to DevSecOps
- Fuzzing
- DevSecOps with Helix++
- Research Challenges and Resources

Security Vulnerabilities 2019



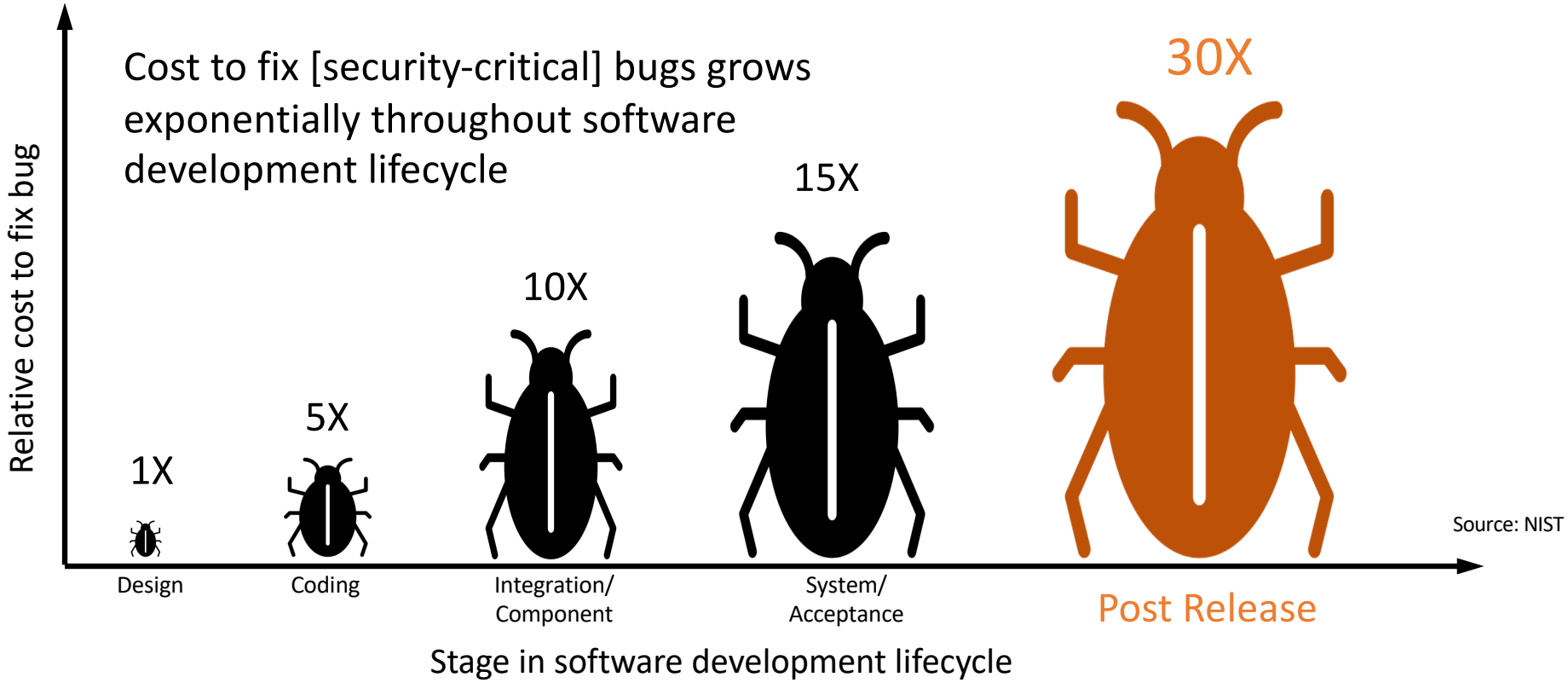
Source: CVE details / NIST National Vulnerability Database

Even if your software is secure, other software is not

Infrastructure and libraries are **not** immune!

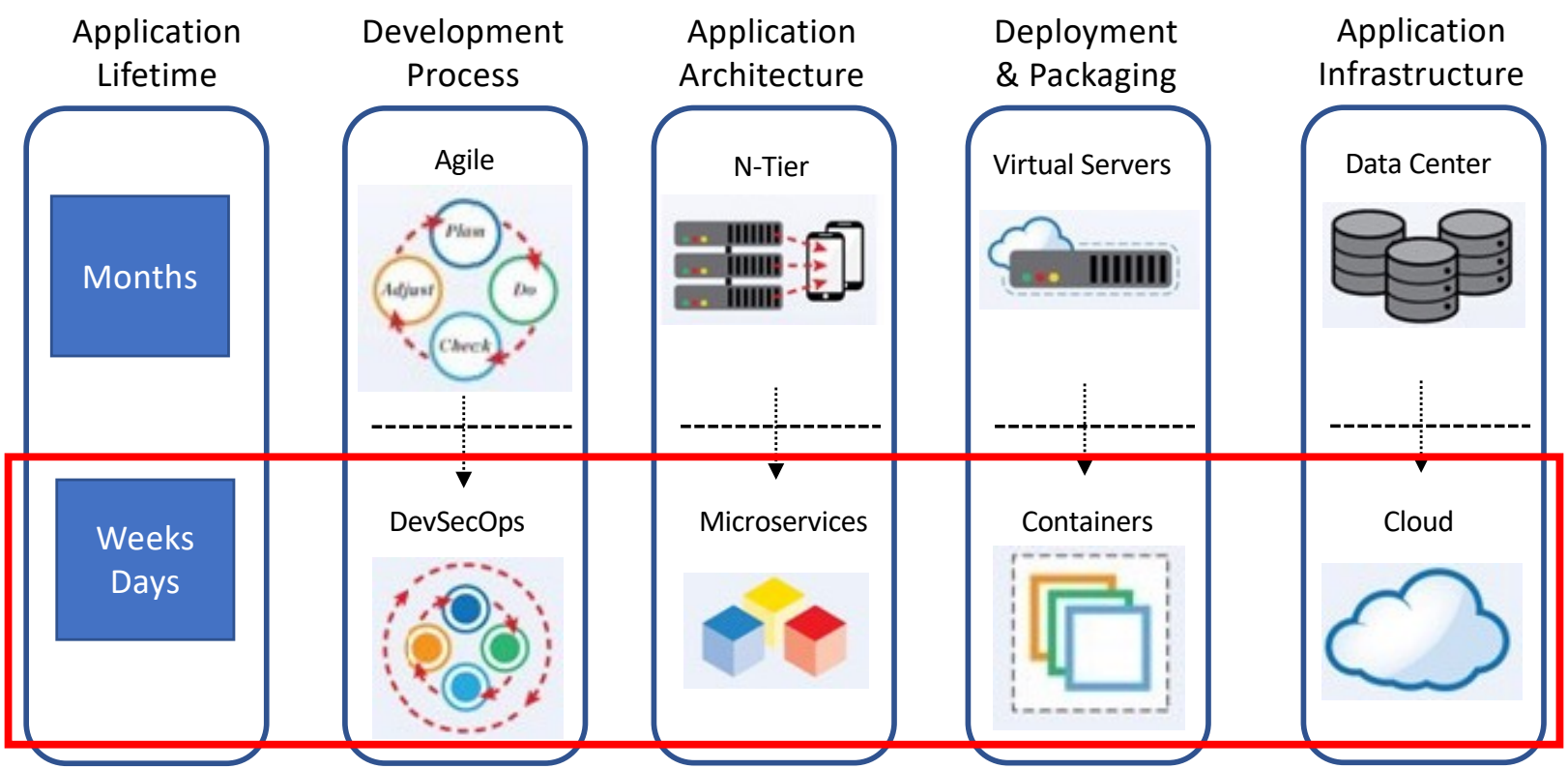
- Web, name, authentication servers
- Anti-virus software
- Commonly used critical libraries, e.g. openSSL, libC,
- Remote desktop

Problem: Traditional DevOps → Fragile Software



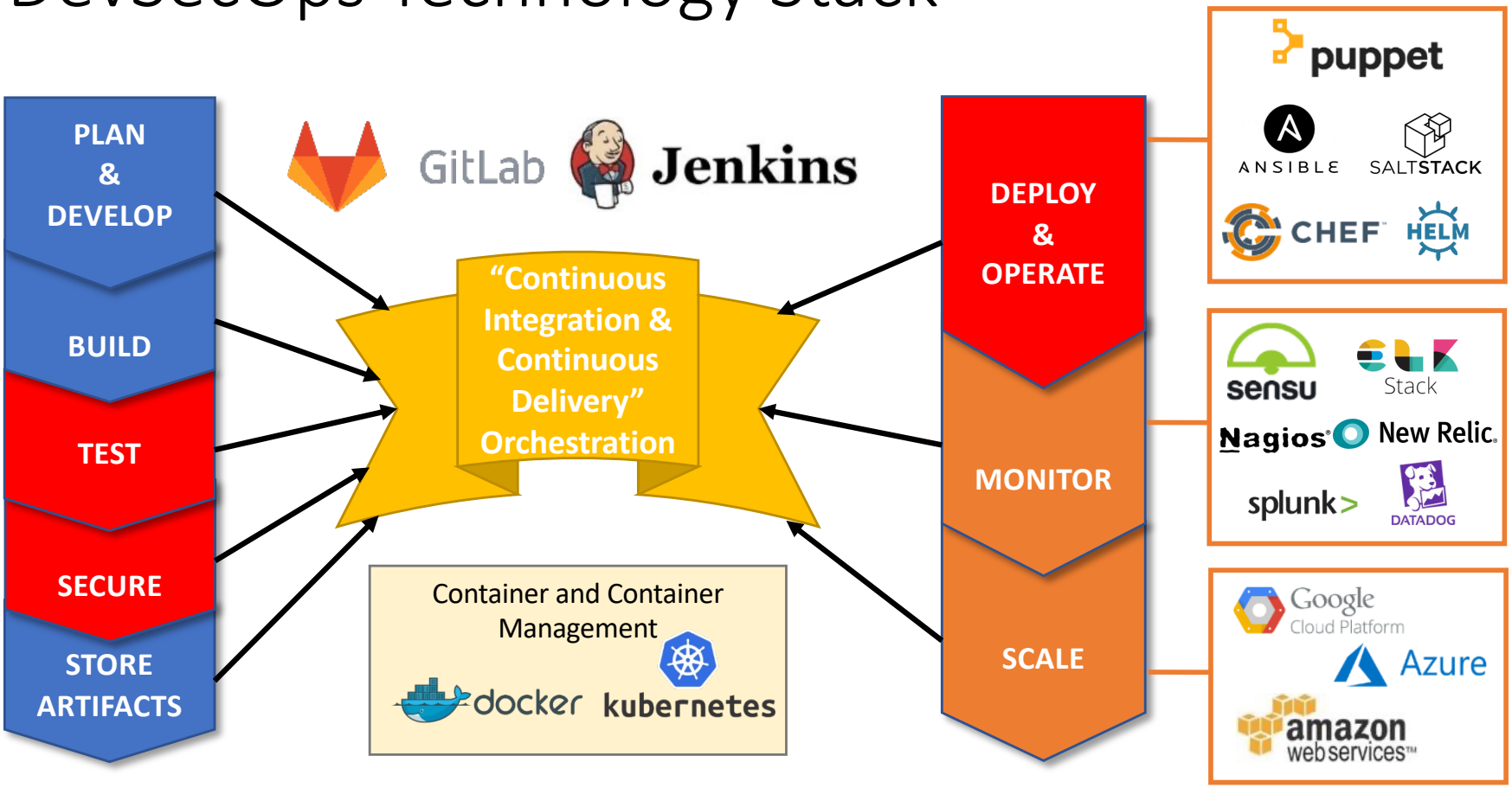
UNIVERSITY of VIRGINIA | SCHOOL of ENGINEERING & APPLIED SCIENCE

DevOps to DevSecOps



Credit: Nicholas Chaillan OSD

DevSecOps Technology Stack



Credit: Nicholas Chaillan OSD

Outline of Talk

- Brief History of Software Development Methodologies
  - Waterfall
  - Agile
  - DevOps
- Left Shift from DevOps to DevSecOps
- Fuzzing
- DevSecOps with Helix++
- Research Challenges and Resources

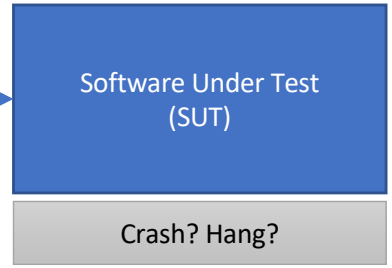
Software Fuzz-testing

fuzz /fʌz/

n. 1 data that has been fuzzed; data that has been **mutated**, **malformed**, **invalid** data

v. 2 to fuzz; to input/feed **malformed** data into a hardware or software system with the intention of crashing the system and **exposing exploitable vulnerabilities**

```
14hkj3nkjlvh9343434324jdjjjjjj
@@@P'tain chuis beau! @@@
91^*&^(DF34hgaszzz
91y4yvjnskl
J'M le cHoCoLaT
%&*^%*^IUGFG
To be or not to be
AAAAAAAAAAAAAAAAAA
01010001010100101
```



An Empirical Study of the Reliability of Unix Studies, Barton T. Miller '89

Software Fuzz-testing

- Today's leading **automated bug-finding** approach
- **Uncover bugs** by bombarding program with inputs
- **Coverage-guided search**: breed only the **winners**
  - Measure each input's code coverage via tracing
  - Keep and mutate only those reaching new code

Source: S. Nagy

Cyber Grand Challenge Competition

- 2014 Competition to build cyber reasoning system.
- Competitors presented with Challenge Binaries (CBs)
- Had to find vulnerabilities, prove existence of vulnerabilities, mitigate them, and weaponize them
- Of the seven finalists, six independently chose to use fuzzing to find vulnerabilities

Why Binary-only Fuzzing?

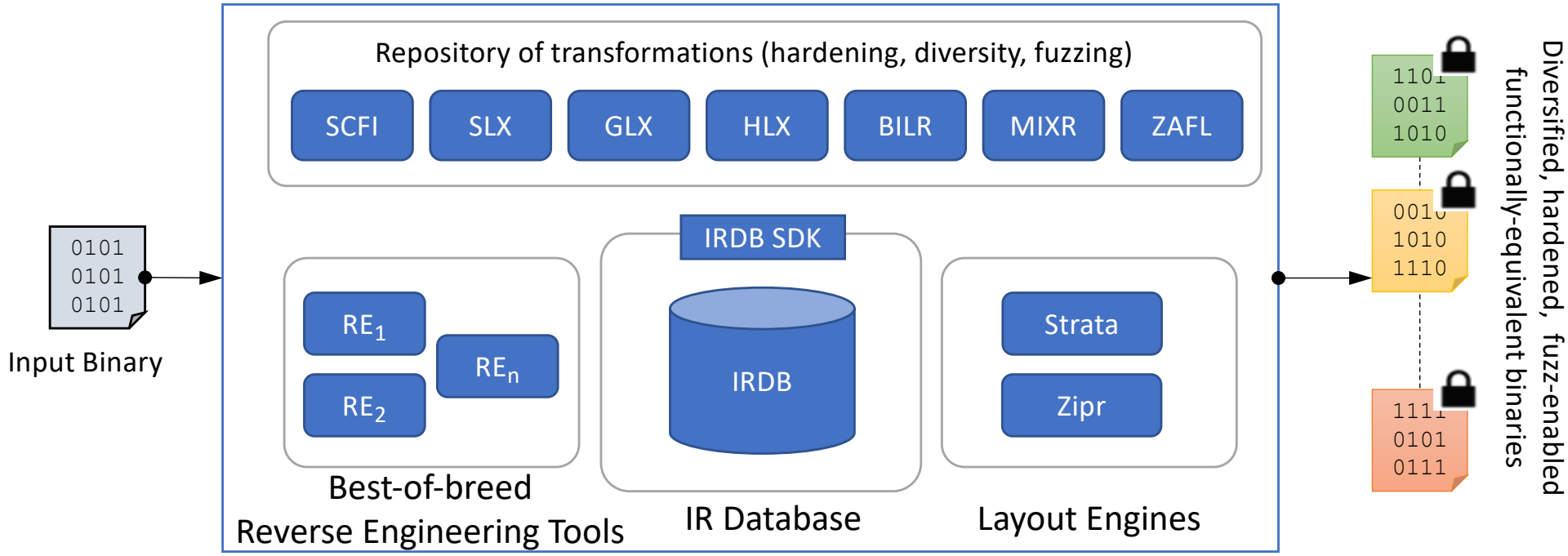
Advantages

- Source code often not available
  - Closed source
  - Third-party libraries
- Multi-language systems not supported by all fuzzing tools
  - Legacy (Assembly, Ada, Fortran, etc.)
  - Emerging languages (Rust, Go, etc.)
- Build systems are complex and fragile
  - Difficult to modify build system
  - Specific software tools required to build application
- WYSINWYX – What you see is not what you execute

Disadvantages

- Quality (speed, coverage, etc.) can suffer. Depends on reverse engineering tools

Helix++ Binary Rewriting Infrastructure

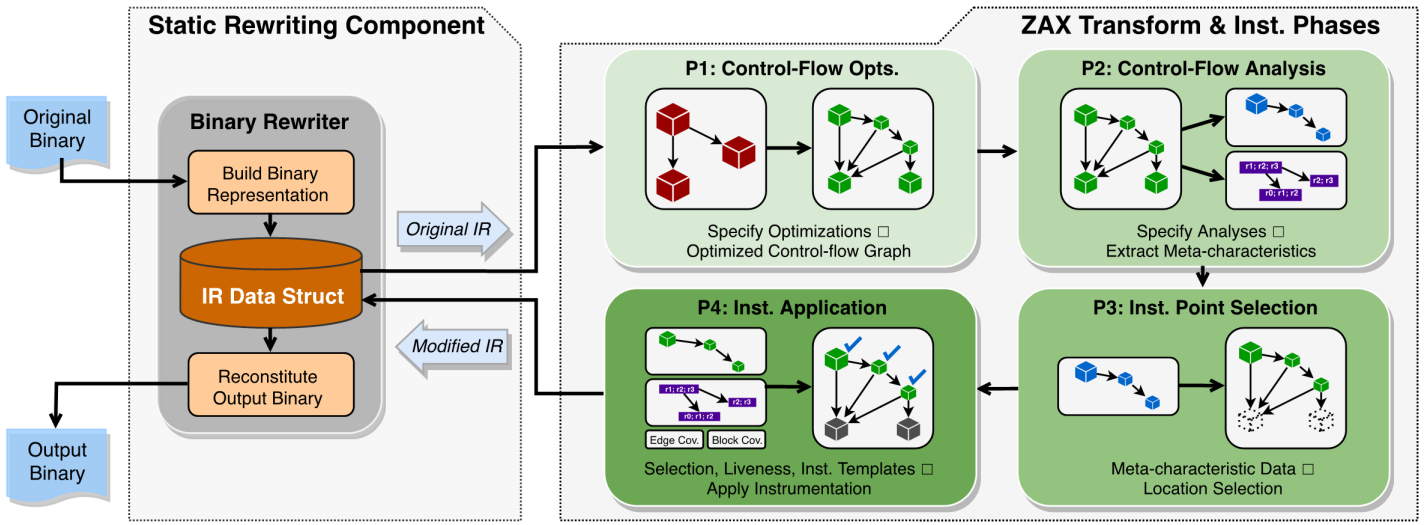


- No source code required, no vendor cooperation needed
- Helix key technology in DARPA Cyber Grand Challenge
  - #2 overall, #1 in efficient defenses
- Applied to wide variety of software
- Network facing daemons, drone software, large DOD applications, GUIs, desktop/laptop utilities

UNIVERSITY of VIRGINIA | SCHOOL of ENGINEERING & APPLIED SCIENCE

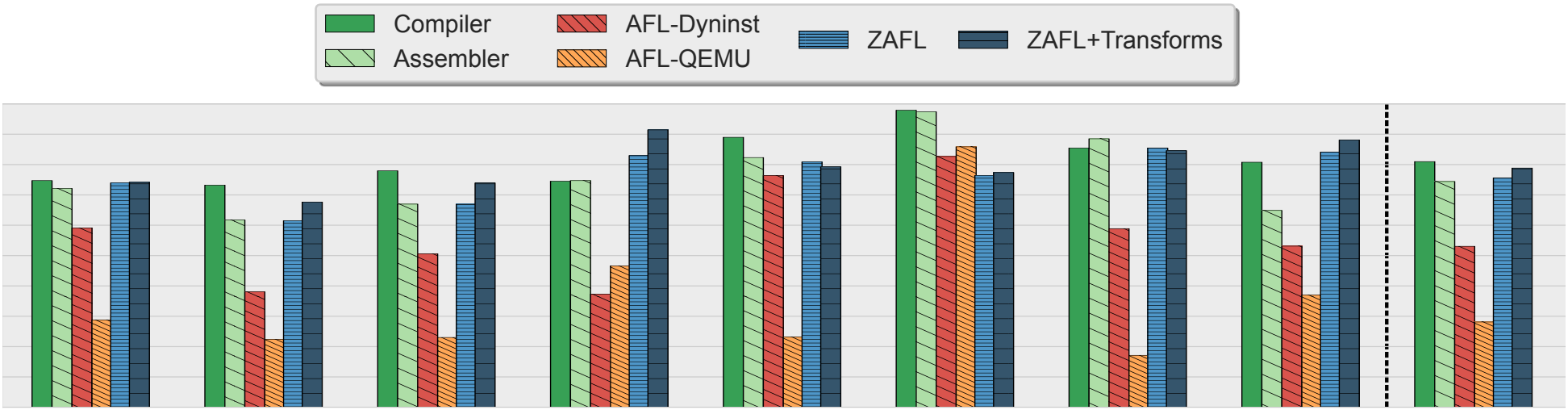
The Helix ++ ZAFL Platform – Fast Binary-only Fuzzing

- **Statically**-inserted, **inlined** instrumentation with **liveness awareness**



Same Coverage, Less Bloat: Accelerating Binary-only Fuzzing with Coverage-preserving Coverage-guided Tracing, S. Nagy, A. Nguyen-Tuong, J. Hiser, J. Davidson, M. Hicks, *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*.

ZAFL Speed Comparison



Compiler: **24%**, Assembler: **34%**
AFL-Dyninst: **88%**, AFL-QEMU: **256%**
ZAFL: **32%**, ZAFL+Transforms: **27%**

Comparative Evaluation of Binary Rewriters

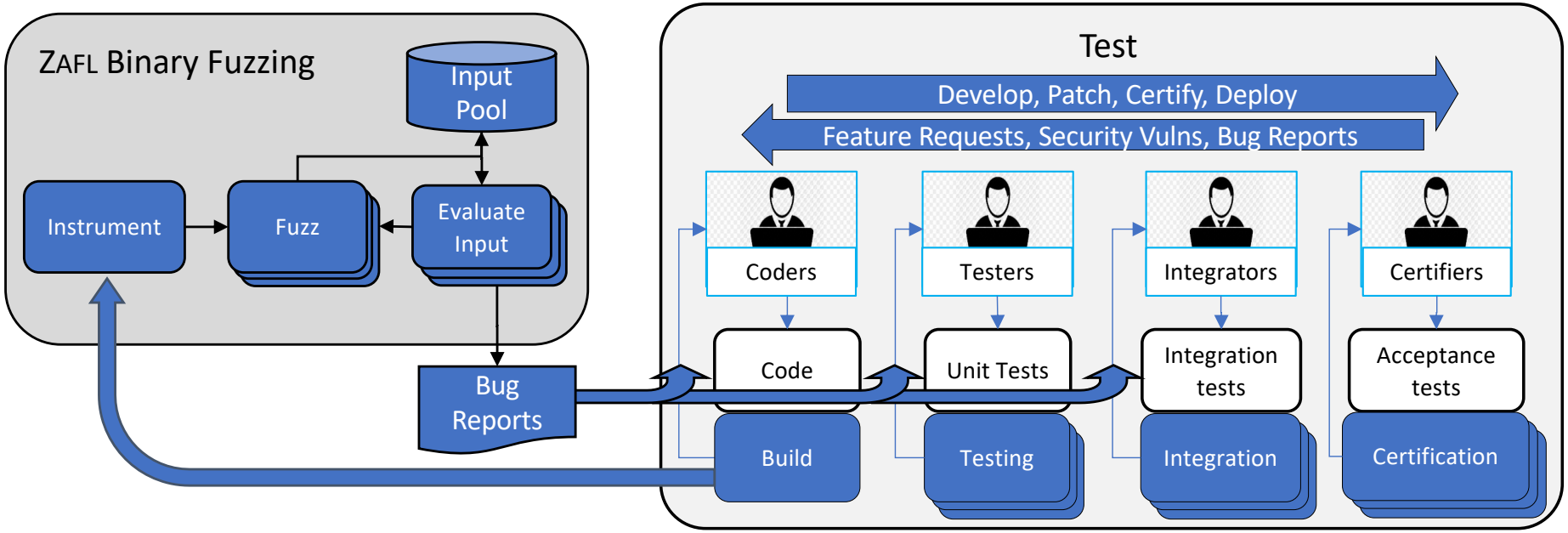
Tool	Null EXE	Null Func
ddiasm	88.97%	85.91%
e9patch	100.00%	78.34%
egalito	98.50%	29.39%
mctoll	0.89%	0.89%
multiverse	26.31%	10.82%
reopt	76.43%	33.91%
retrowrite	9.98%	9.24%
revng	26.46%	23.50%
uroboros	6.45%	2.87%
zipr	100.00%	100.00%

Source: A Broad Comparative Evaluation of X86-64 Binary Rewriters, E. Schulte, M. Brown, V. Folts, CSET 2022

Outline of Talk

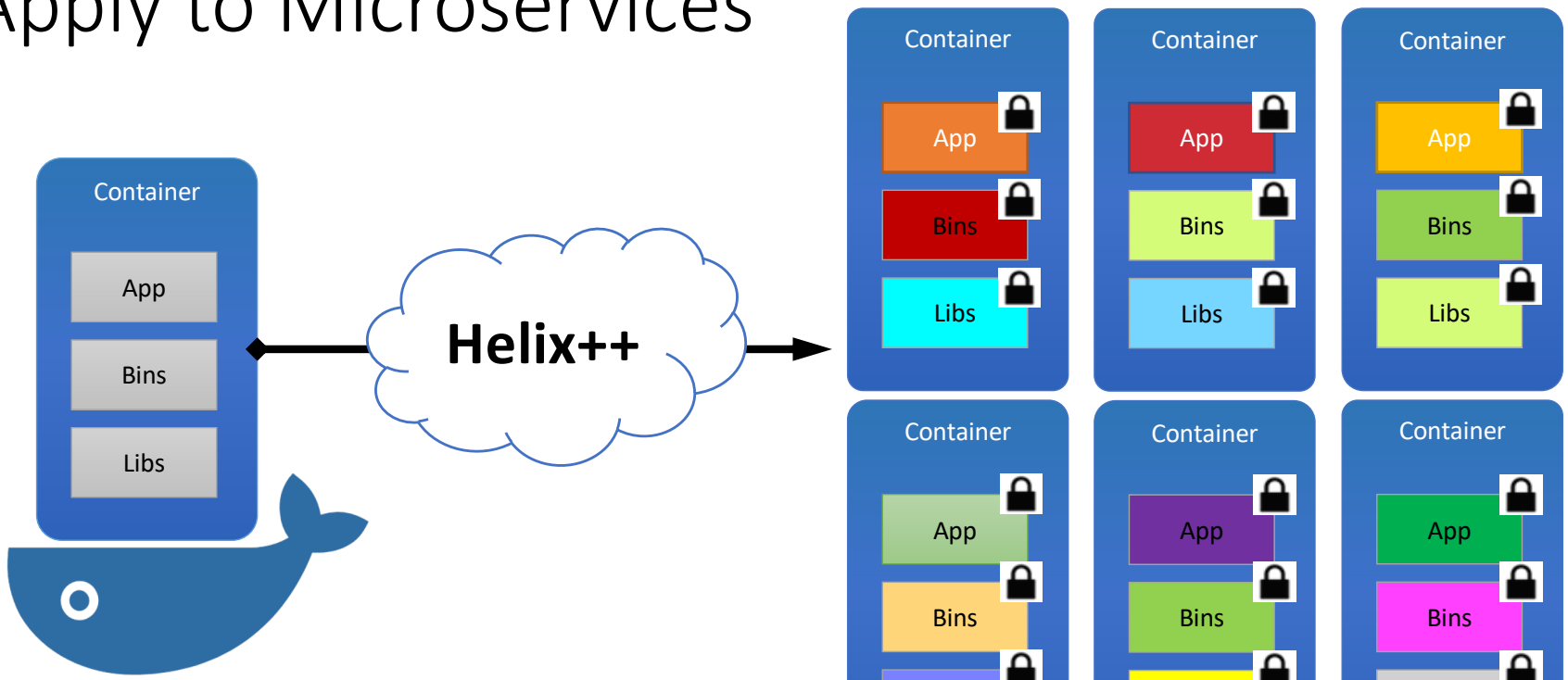
- Brief History of Software Development Methodologies
  - Waterfall
  - Agile
  - DevOps
- Left Shift from DevOps to DevSecOps
- Fuzzing
- DevSecOps with Helix++
- Research Challenges and Resources

DevOps ➔ DevSecOps



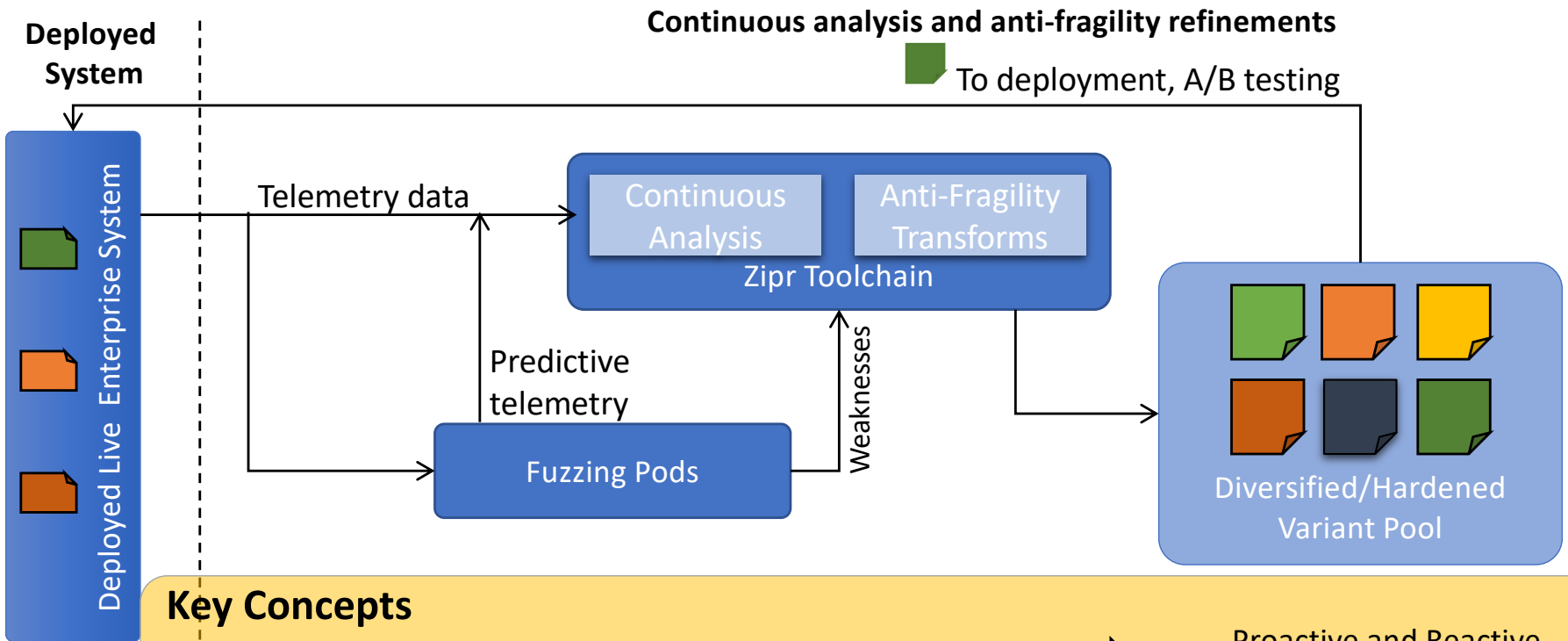
UNIVERSITY of VIRGINIA | SCHOOL of ENGINEERING & APPLIED SCIENCE

Apply to Microservices



Helix++ enables the production of **hardened** and **diversified** container images

Continuous and Proactive MTD



Key Concepts

Continuous analysis to refine understanding

Continuous fuzzing to identify weaknesses

➡ Proactive and Reactive Improvements

Outline of Talk

- Brief History of Software Development Methodologies
  - Waterfall
  - Agile
  - DevOps
- Left Shift from DevOps to DevSecOps
- Fuzzing
- DevSecOps with Helix++
- Research Challenges and Resources

DevSecOps Research Questions

- What are the metrics for CI fuzzing? Are there different metrics besides coverage?
- Use fuzzing to enhance test suites. Effective or not?
- Leverage production environment to enhance CI fuzzing. Continuous and proactive, reactive improvements (A/B testing)
- Is a Moving Target Defense (MTD) feasible and effective at the microservices level?
- Ensemble fuzzing: Leverage different fuzzing approaches (mutation, coverage metrics, etc.)
- Specialized, targeted fuzzer to test patches
- Differential testing across different build configurations

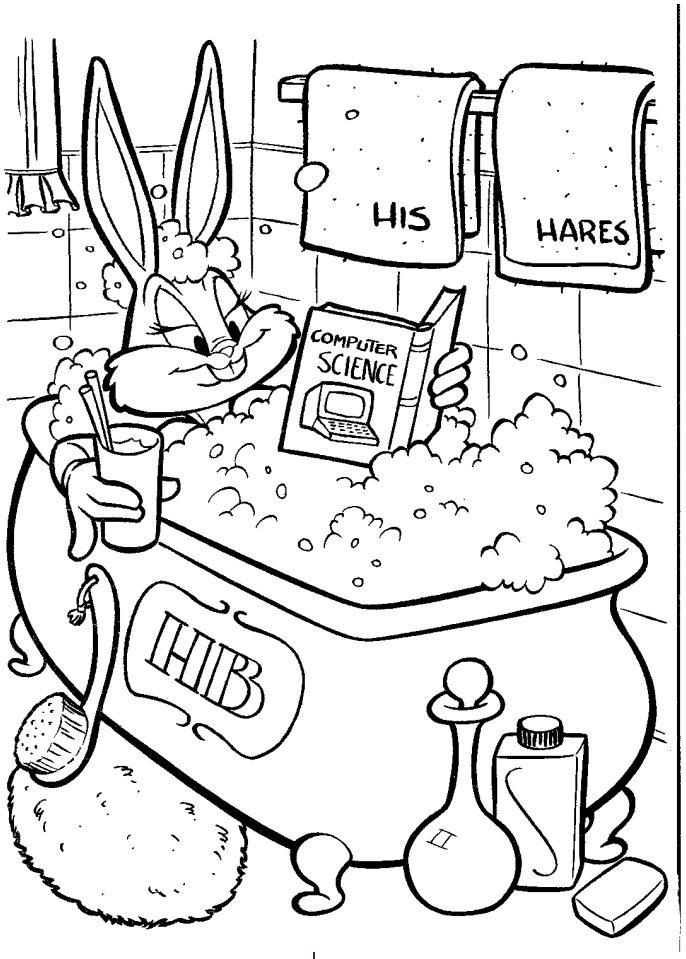
Resources

- Zafl fuzzer is open source
  - https://git.zephyr-software.com/opensrc/libzafl/-/wikis/Home
- Turbo (CI/CD) pipeline tools are open source
  - https://git.zephyr-software.com/opensrc/turbo
- Same Coverage, Less Bloat: Accelerating Binary-Only Fuzzing with Coverage-Preserving Coverage-Guided Tracing, S. Nagy, A. Nguyen-Tuong, J. Hiser, J. Davidson, M. Hicks, CCS 2021, https://doi.org/10.1145/3460120.3484787
- Breaking Through Binaries: Compiler-quality Instrumentation for Better Binary-only Fuzzing, S. Nagy, A. Nguyen-Tuong, J. Hiser, J. Davidson, M. Hicks, USENIX 2021, https://www.usenix.org/system/files/sec21-nagy.pdf
- A Broad Comparison of x86-64 Binary Rewriters, E. Schulte, M. Brown, V. Folts, CSET 2021, https://doi.org/10.1145/3546096.3546112
- Fuzzing: A Survey for Roadmap, X. Zhu, S. Wen, S. Camptepe, Yang Xiang, Computing Surveys, https://doi.org/10.1145/3512345

Acknowledgments

- This material is based upon work supported by the National Science Foundation under
  - Grant No. 2115130—Helix++: Securing Open Science Platforms
  - DARPA CGC, Contract No. FA8750-14-C-0110, and
  - DARPA CFAR Contract No. FA8750-15-C-0118
- Thanks to Stefan Nagy for slides and graphs
- Thanks to Jason Hiser and Anh Nguyen-Tuong for slides and advice

Questions



UNIVERSITY of VIRGINIA | SCHOOL of ENGINEERING & APPLIED SCIENCE