Scratching the Surface of `./configure`: Learning the Effects of Compile-Time Options on Binary Size and Gadgets

Xhevahire Tërnava[1], Mathieu Acher[12], Luc Lesoil[1], Arnaud Blouin[13], Jean-Marc Jézéquel[1]

[1]Univ Rennes, CNRS, Inria, IRISA - UMR 6074, F-35000 Rennes

[2]Institut Universitaire de France (IUF)

[3]INSA Rennes

ICSR'22 – June 15-17, 2022

Context

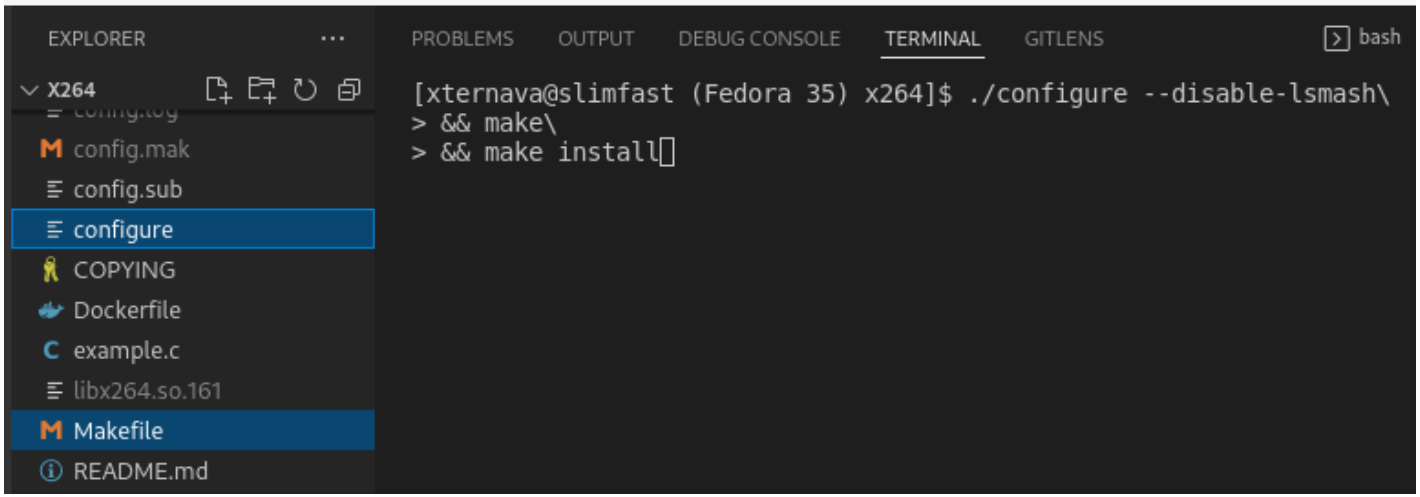
- Modern software systems are highly configurable, all powered by a large number of configuration options
- The compile-time options are extensive in C-based systems (set during `./configure`)
  - There is hardly any work on their impact on the system's non-functional properties (binary size and attack surface)
  - Should users change the system's default configuration?

./configure and gadgets

Background and motivation

./configure flavour

- It helps to customize a given C-based software system
- *e.g.*, this disables the MP4 support in 264 video encoder

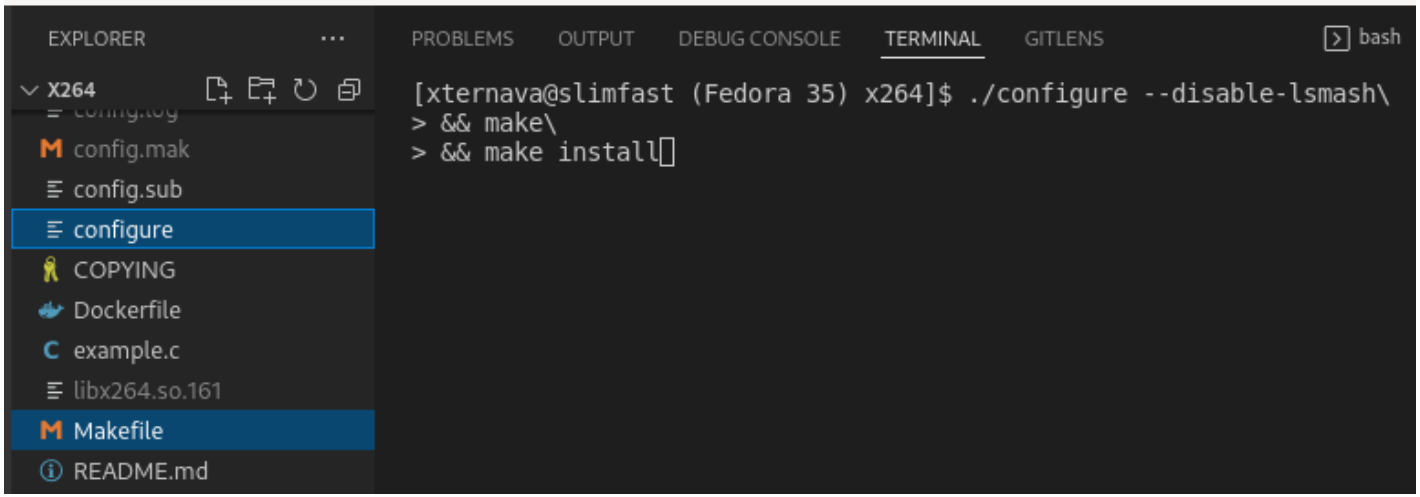


./configure and gadgets

Background and motivation

./configure flavour

- It helps to customize a given C-based software system
- *e.g.*, this disables the MP4 support in 264 video encoder



- Issues:
  - Sys. admins often make poor configuration choices, *e.g.*,
    - Default Hadoop results in the worst possible performance [1]
    - "*The size of SQLite can be reduced below 180KiB*" [2]
  - End-users lack the configuration knowledge

./configure and **gadgets**

Background and motivation

Code reuse gadgets (RET or JMP instructions)

- The security of modern software systems is threaten internally
- Small code sequences that are chained to threaten the system
- Security metric: number of gadgets that can be exploited

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   GITLENS                bash

0x0000000000413cd0 : xor r8d, r8d ; mov esi, 0x5dfa36 ; call qword ptr [rbx + 0x58]
0x0000000000408d50 : xor r8d, r8d ; mov rax, r8 ; ret
0x0000000000495bcc : xor r8d, r8d ; xor edx, edx ; jmp 0x495ae0
0x0000000000529bbc : xor r8d, r8d ; xor edx, edx ; jmp 0x529ac6
0x00000000004456db : xor r8d, r8d ; xor esi, esi ; jmp 0x445708
0x000000000042d141 : xor r8d, r8d ; xor esi, esi ; mov rax, r12 ; jmp 0x42d170
0x00000000004a44d9 : xor r8d, r8d ; xor esi, esi ; xor ecx, ecx ; jmp 0x4a46dd
0x0000000000537bd9 : xor r8d, r8d ; xor esi, esi ; xor ecx, ecx ; jmp 0x537ddd
```

./configure and **gadgets**

Background and motivation

Code reuse gadgets (RET or JMP instructions)

- The security of modern software systems is threaten internally
- Small code sequences that are chained to threaten the system
- Security metric: number of gadgets that can be exploited

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   GITLENS                bash

0x0000000000413cd0 : xor r8d, r8d ; mov esi, 0x5dfa36 ; call qword ptr [rbx + 0x58]
0x0000000000408d50 : xor r8d, r8d ; mov rax, r8 ; ret
0x0000000000495bcc : xor r8d, r8d ; xor edx, edx ; jmp 0x495ae0
0x0000000000529bbc : xor r8d, r8d ; xor edx, edx ; jmp 0x529ac6
0x00000000004456db : xor r8d, r8d ; xor esi, esi ; jmp 0x445708
0x000000000042d141 : xor r8d, r8d ; xor esi, esi ; mov rax, r12 ; jmp 0x42d170
0x00000000004a44d9 : xor r8d, r8d ; xor esi, esi ; xor ecx, ecx ; jmp 0x4a46dd
0x0000000000537bd9 : xor r8d, r8d ; xor esi, esi ; xor ecx, ecx ; jmp 0x537ddd
```

- Issues:
  - smaller binary size $\longrightarrow$ fewer gadgets (?)
  - Lack of knowledge how much an option threaten a system
  - What is the effect of options on the system's attack surface?

Study Design

– Goal and Research Questions –

Goal: To quantify and learn the effects of compile-time options on binary size and attack surface of C-based software systems

RQ_1: What is the effect of compile-time options of a system on its binary size?

RQ_2: What is the effect of compile-time options of a system on its attack surface (*a.k.a.*, gadgets)?

RQ_2: Which compile-time options are the most influential on the binary size and the gadgets of a software system?

Study Design

– Subject Systems –

- We analysed 4 C-based open-source projects
  - 86 – 15.6k ⋆⋆⋆, 1.3k – 23.7k git, 17-94 

Subject	$\#Options$	Baseline B. Size	Baseline # Gadgets
`x264`	25/39	3,096,112	106,878
`nginx`	91/127	4,507,168	29,925
`SQLite`	31/72	8,561,208	67,734
`xz`	36/88	1,254,536	9,121

** Baseline configuration ≡ Default configuration (in git repository)

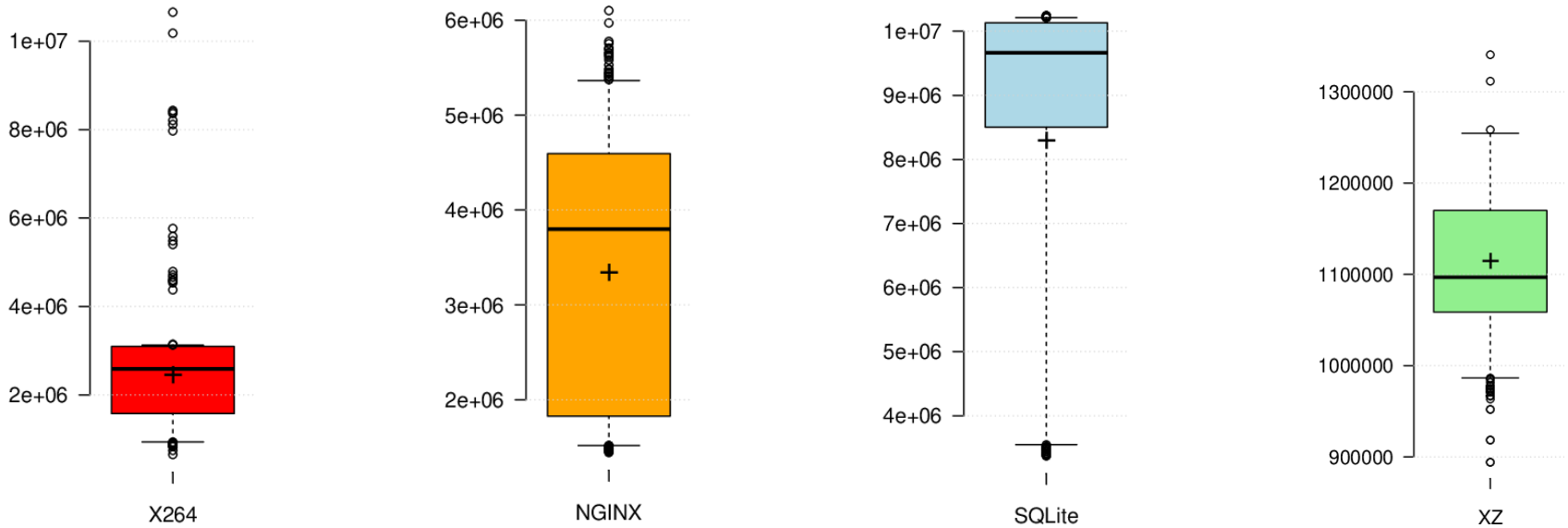
Study Design

– The Conducted Experiment –

1. Compiled each subject in its default configurations
2. Explored its configuration space and build its Feature Model
3. Customized each subject, i.e., >400 random configurations
4. Identify most influential options in non-functional properties

The effect of compile-time options on binary size

Results of RQ_1



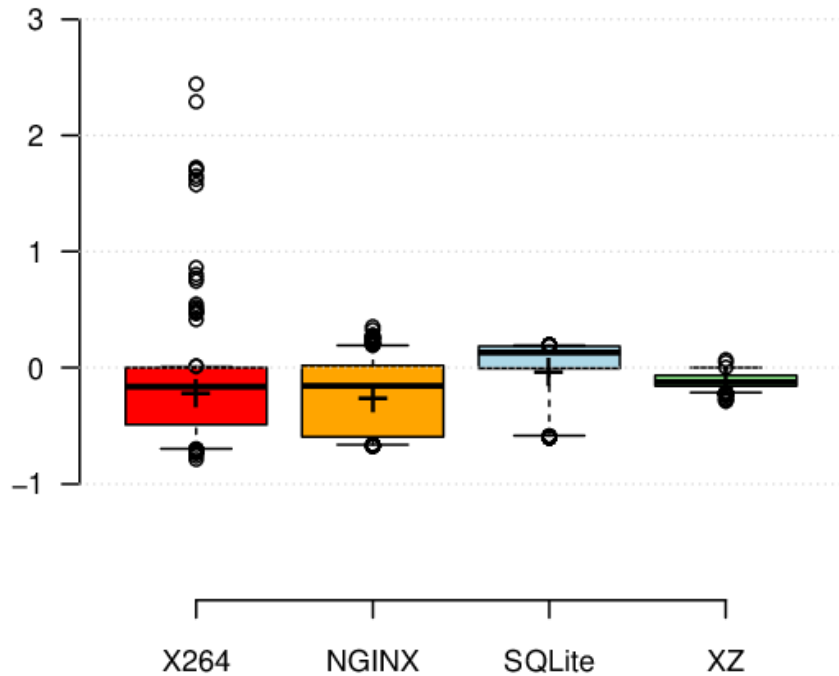
- `x264`: 0.62 MiB – 10.16 MiB
- `nginx`: 1.38 MiB – 5.82 MiB
- `SQLite`: 3.22 MiB – 9.77 MiB
- `xz`: 0.85 MiB – 1.28 MiB

The effect of compile-time options on binary size

Results of RQ_1

Comparing with the baseline binary size (0% in the figure):

- Larger binary size: 26% of the configurations
- Same binary size: 13% of the configurations
- Smaller binary size: 61% of the configurations

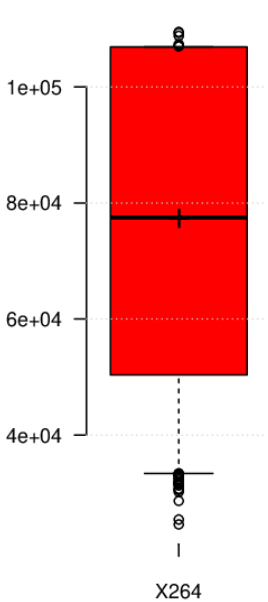


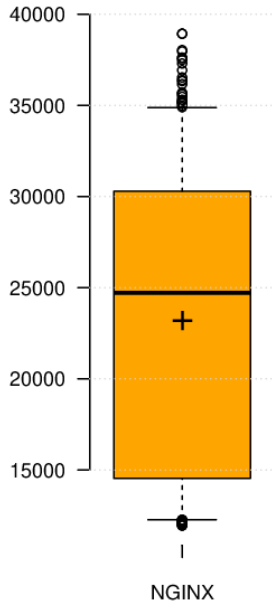
BS decreases for more bytes (36%) than that it increases (33%)

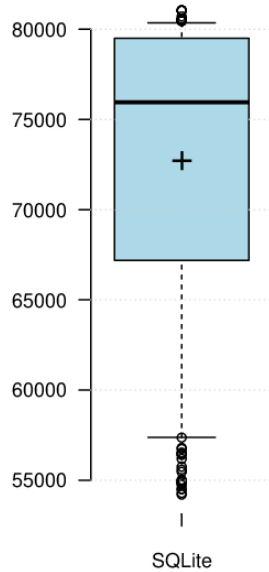
$\longrightarrow$ The compile-time customization of a software system has mainly a positive effect on its binary.

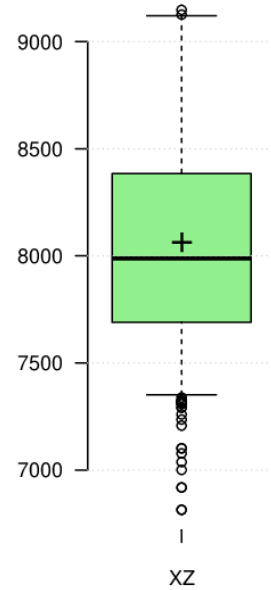
The effect of compile-time options on attack surface

Results of RQ_2









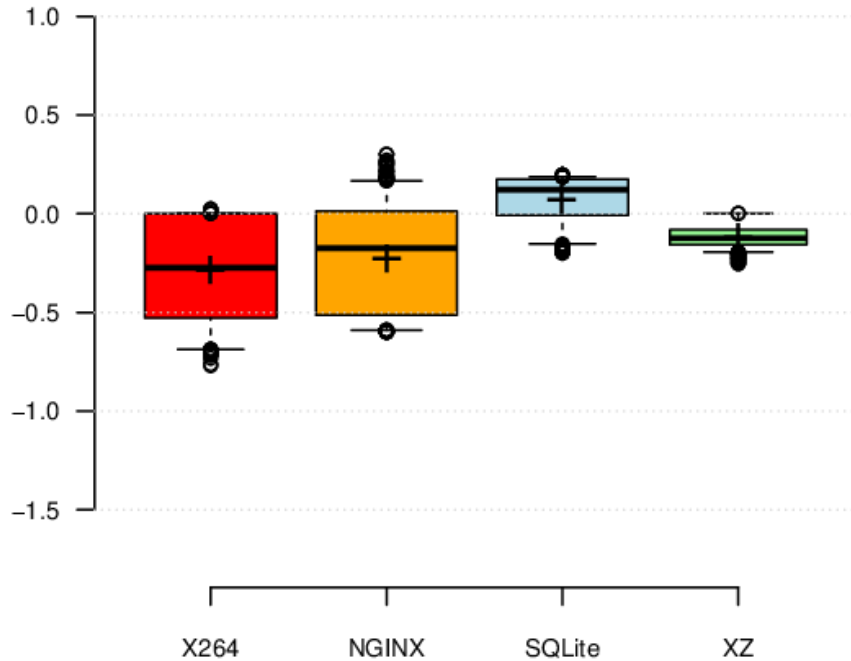
- `x264`: 25K – 109K gadgets
- `nginx`: 12K – 39K gadgets
- `SQLite`: 54K – 81K gadgets
- `xz`: 7K – 9K gadgets

The effect of compile-time options on gadgets

Results of RQ_2

Comparing with the baseline # gadgets (0% in the figure):

- Larger # gadgets: 25% of the configurations
- Same # gadgets: 12% of the configurations
- Smaller # gadgets: 63% of the configurations



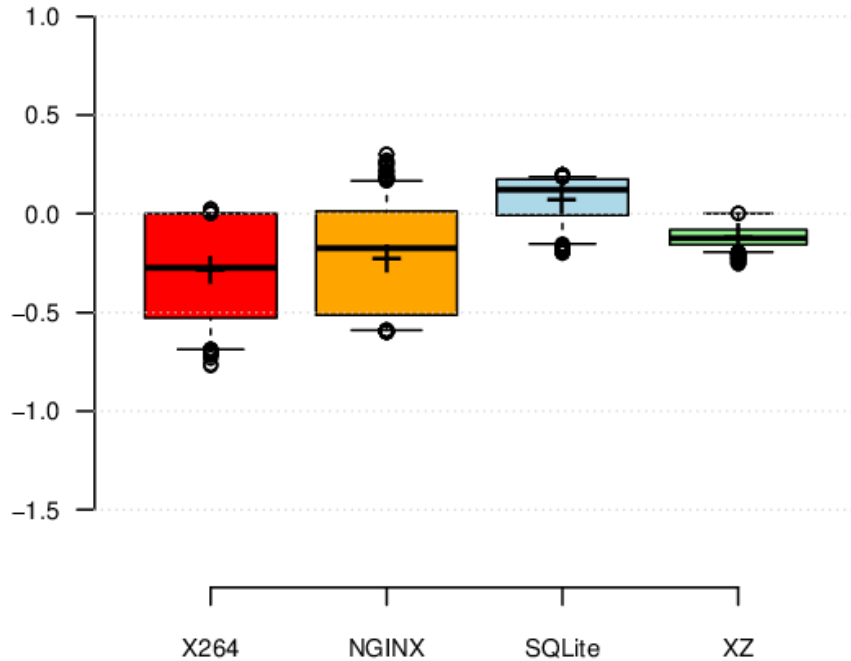
On average, the attack surface will be reduced far more (25%) than that it will be increased (6%)

The effect of compile-time options on gadgets

Results of RQ_2

Comparing with the baseline # gadgets (0% in the figure):

- Larger # gadgets: 25% of the configurations
- Same # gadgets: 12% of the configurations
- Smaller # gadgets: 63% of the configurations



On average, the attack surface will be reduced far more (25%) than that it will be increased (6%)

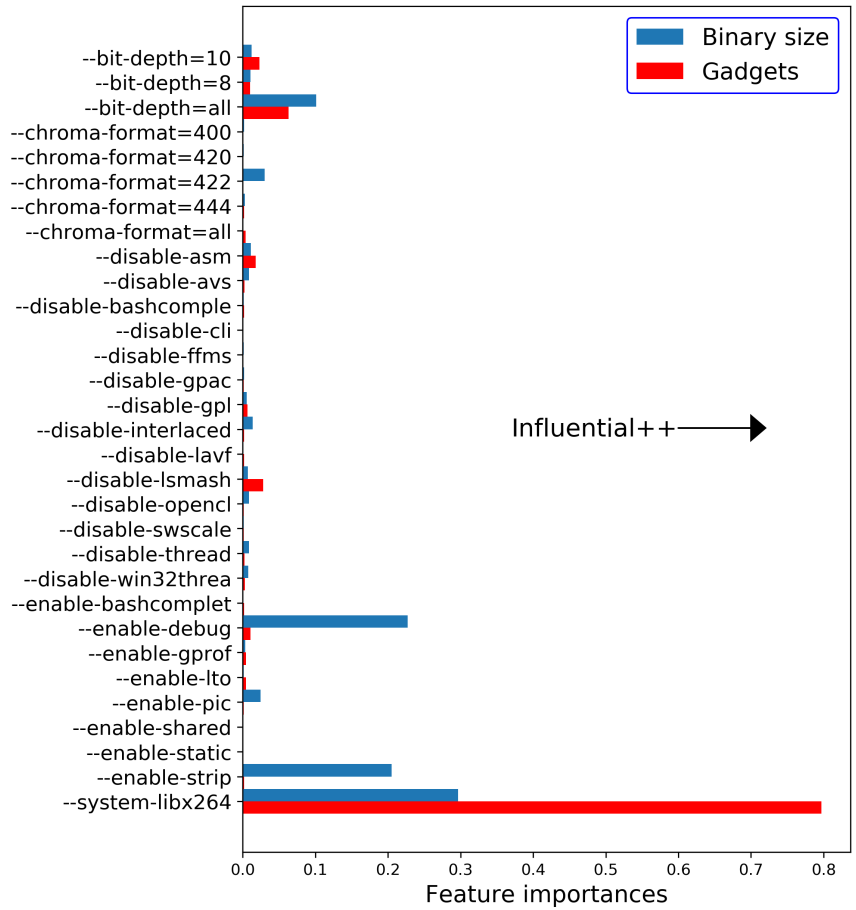
There is a weak (0.52) to almost perfect (0.99) Pearson correlation between the variation of binary size and # gadgets in a system

Influential compile-time options

Results of RQ_3

Which options are the most influential in these systems?

- x264: –system–libx264
  0.30 for binary size
  0.80 for gadgets
- nginx: –without-http
  0.85 feature importance
- SQLite: –enable-all and –enable-fts5
  0.90 feature importance
- xz: importance is spread
  0.17 for binary size
  0.24 for gadgets

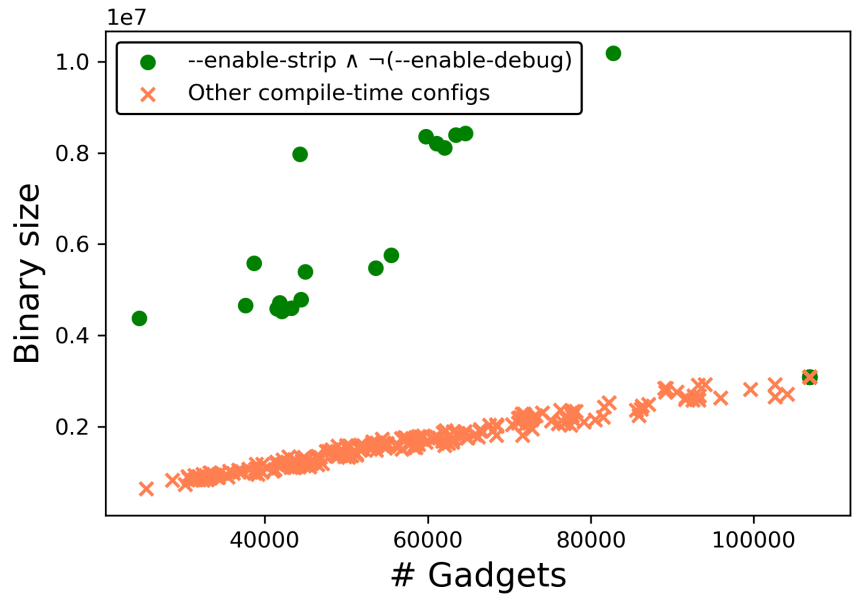


Influential compile-time options

Results of RQ_3

Are the influential options of a system the same for binary size and gadgets?

- x264: *e.g.*, –system–libx264
  (0.18) for binary size
  (0.78) for gadgets
- nginx: are the same
- SQLite: are the same
- xz: *e.g.*, –enable-checks
  (0.17) for binary size
  (0.05) for gadgets



The influence of an option on binary size and attack surface differs

Summary: **Scratching the Surface of** `./configure`

- An empirical evaluation about the effects of compile-time options on the system's non-functional properties
  - The compile-time customization of a software system has mainly a positive effect on its binary size and attack surface
    - the system's binary size varies between -79% and 244%
    - the system's attack surface varies between -77% and 30%
  - Variation of gadgets has a week (0.52) to almost perfect (0.99) correlation to the binary size of a system
  - Developers and integrators can use prediction models to take informed decisions when configuring a system
- Costs: Human cost, Time and Disk resources, Learning cost
- Benefits: Knowing the effects on non-functional properties, Identification of influential options (and their interactions)
- The availability of the experiment and artifacts:
  Git: `https://github.com/diverse-project/confsurface`
  Zenodo: `https://doi.org/10.5281/zenodo.6401250`