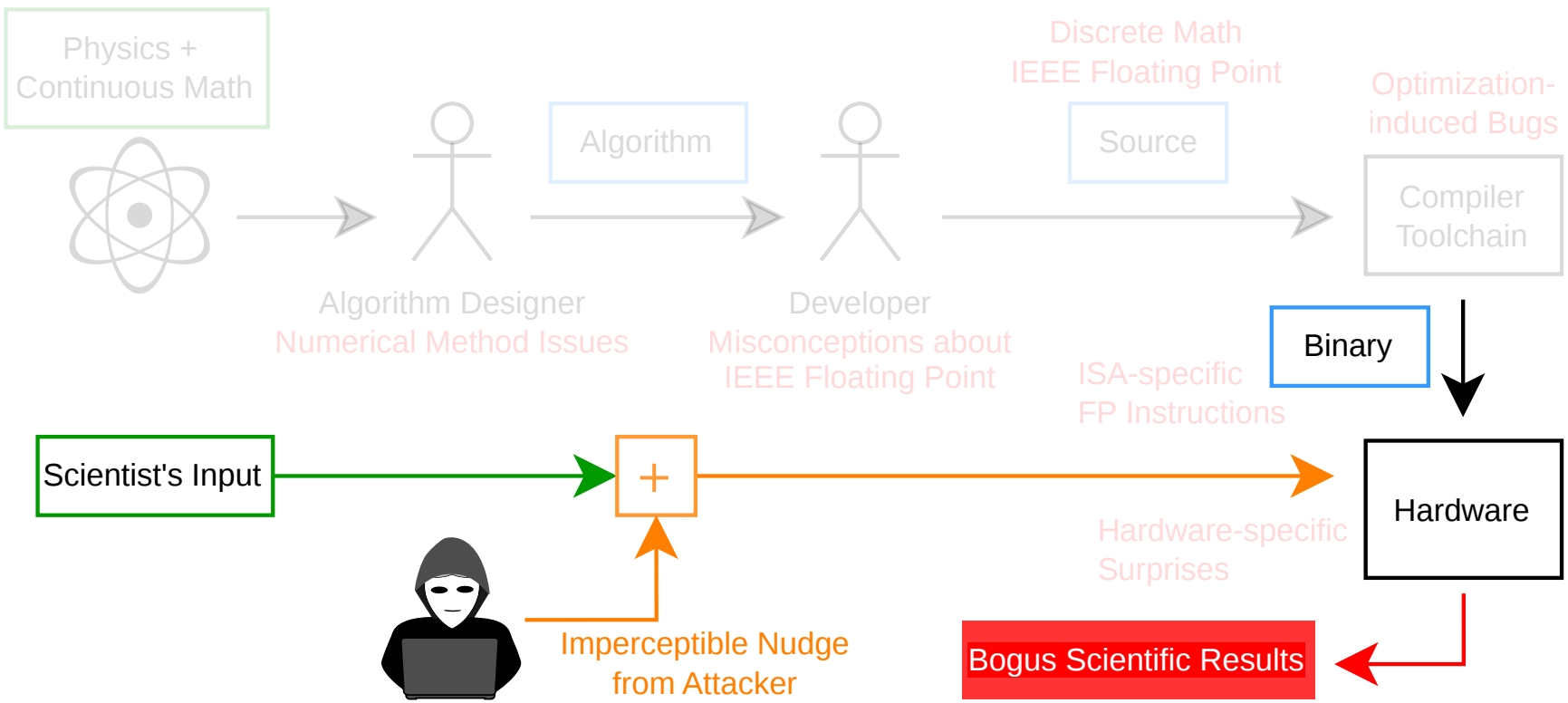
Exploring A Novel Nudging Attack on Scientific Applications

Michael Polinski
Peter Dinda

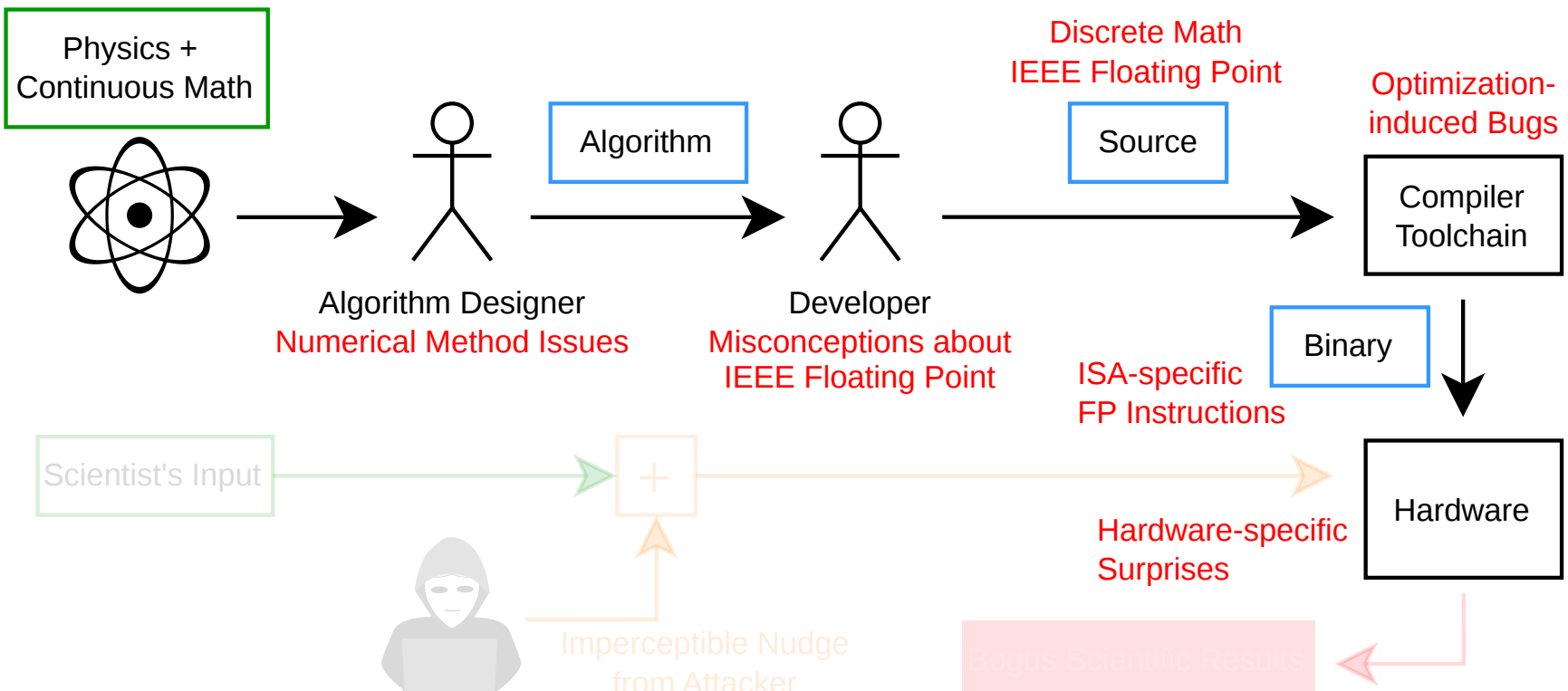
Summary

- *Scientific software is vulnerable to conventional attacks (our work)*
- **Scientific software is vulnerable to new forms of attack**
- Our new "**nudging attack**" involves only small, low-perceptibility manipulation of input numbers via **fuzzing**
- Nudging attack works against chaotic dynamical systems
- Nudging attack works against **GROMACS, LAMMPS, Laghos**
  - $\sim 63\text{x}$ output deflection across experiments (**2 hours**)
  - $\leq 10^{-6}$ nudge $\Rightarrow 10^{14}$ output error

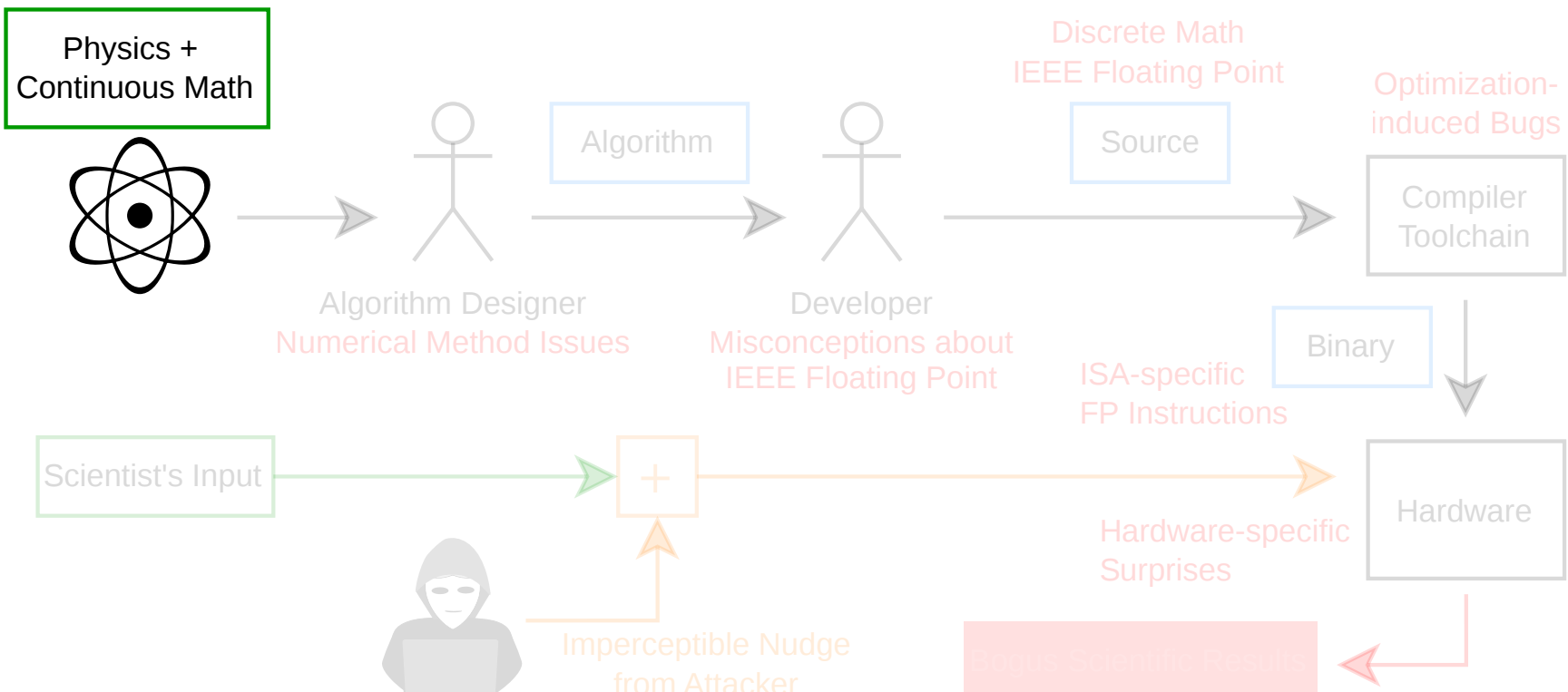
Threat Model



Threat Model



Threat Model



Attacker’s Goals

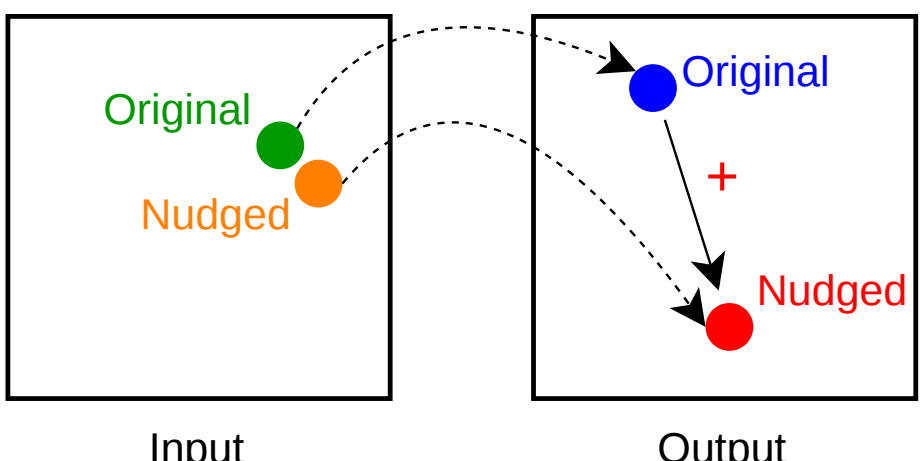
Create bogus yet plausible scientific results

Attacker's Goals

Create bogus yet plausible scientific results

Increase "distance from truth"

Deflect output

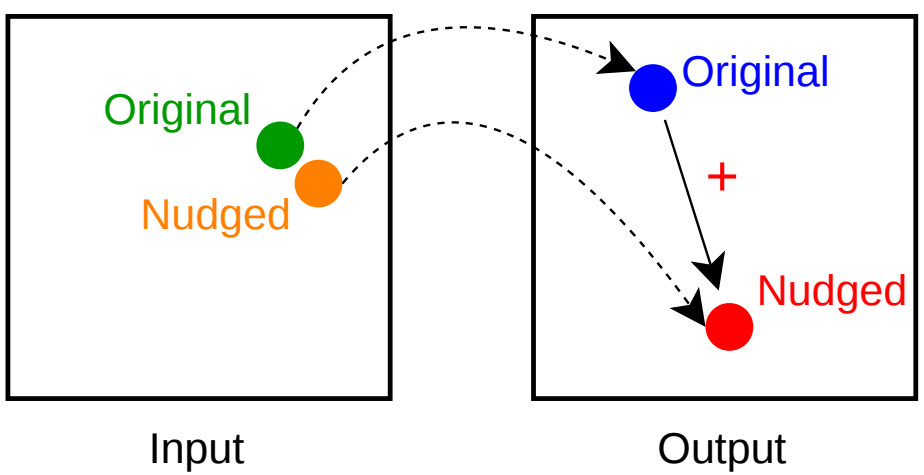


Attacker's Goals

Create bogus yet plausible scientific results

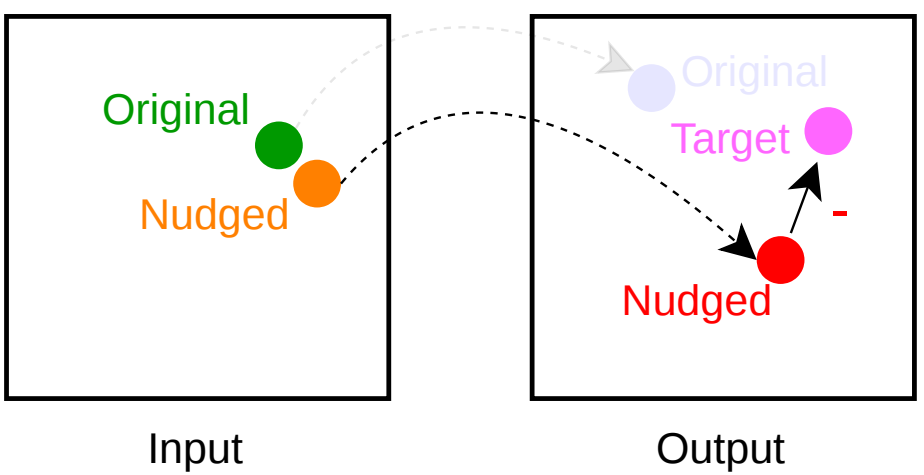
Increase "distance from truth"

Deflect output



Decrease "distance to a target"

Steer output toward a goal



Nudging Attacks

We built a **fuzzer** that finds such “nudging attacks”

Nudging Attacks

We built a **fuzzer** that finds such “nudging attacks”

- **Conventional fuzzing**
  - Software testing technique
  - Repeatedly mutate program input, record behavior
  - Objective: *crash the program*

Nudging Attacks

We built a **fuzzer** that finds such “nudging attacks”

- Conventional fuzzing
  - Software testing technique
  - Repeatedly mutate program input, record behavior
  - Objective: *crash the program*

- **Our use of fuzzing**
  - Fuzzer as a search process
  - Make small input changes
  - Objective: optimize for a *desired output*

Perceptibility Constraints

- A traditional fuzzer searches the whole input space
- The adversary cares about *imperceptible manipulations* to an otherwise legitimate input

```
pair_style  meam/sw/spline
pair_coeff  * * Ti.meam.sw.spline Ti
mass        * 47.90
...
print       "Calculated cohesive energy: ${cohesive_energy} eV/atom"
print       "Reference cohesive energy:  -4.831 eV/atom"
... \\
13
-20.0 0.0
1 0 1 0
1.7426928371636325 46e+00 3.7442771759663893 49e+00 0.0e+00
2.0558017673999966 49e+00 9.1083973090550129 91e-01 1.0870252330109405 39e+01
2.3689106976363607 52e+00 3.8804589663414507 11e-01 -1.5532241883595663 27e+00
2.6820196278727244 10e+00 -1.8840906533469894 27e-02 2.4363004137147852 29e+00
...
```

```
pair_style  meam/sw/spline
pair_coeff  * * Ti.meam.sw.spline Ti
mass        * 47.90
...
print       "Calculated cohesive energy: ${cohesive_energy} eV/atom"
print       "Reference cohesive energy:  -4.831 eV/atom"
... \\
13
-20.0 0.0
1 0 1 0
1.7426928371636325 46e+00 3.7442771759663893 49e+00 0.0e+00
2.0558017673999966 49e+00 9.1083973090550129 91e-01 1.0870252330109405 39e+01
2.3689106976363607 52e+00 3.8804589663414507 11e-01 -1.5532241883595663 27e+00
2.6820196278727244 10e+00 -1.8840906533469894 27e-02 2.4363004137147852 29e+00
...
```

```
pair_style  meam/sw/spline
pair_coeff  * * Ti.meam.sw.spline Ti
mass        * 47.90
...
print       "Calculated cohesive energy: ${cohesive_energy} eV/atom"
print       "Reference cohesive energy:  -4.831 eV/atom"
... \\
13
-20.0 0.0
1 0 1 0
1.7426928371636325 46e+00 3.7442771759663893 49e+00 0.0e+00
2.0558017673999966 49e+00 9.1083973090550129 91e-01 1.0870252330109405 39e+01
2.3689106976363607 52e+00 3.8804589663414507 11e-01 -1.5532241883595663 27e+00
2.6820196278727244 10e+00 -1.8840906533469894 27e-02 2.4363004137147852 29e+00
...
```

```
pair_style  meam/sw/spline
pair_coeff  * * Ti.meam.sw.spline Ti
mass        * 47.90
...
print       "Calculated cohesive energy: ${cohesive_energy} eV/atom"
print       "Reference cohesive energy:  -4.831 eV/atom"
... \\
13
-20.0 0.0
1 0 1 0
1.74269283716363254 6e+00 3.74427717596638934 9e+00 0.0e+00
2.05580176739999664 9e+00 9.10839730905501299 1e-01 1.08702523301094053 9e+01
2.36891069763636075 2e+00 3.88045896634145071 1e-01 -1.55322418835956632 7e+00
2.68201962787272441 0e+00 -1.88409065334698942 7e-02 2.43630041371478522 9e+00
...
```

```
pair_style  meam/sw/spline
pair_coeff  * * Ti.meam.sw.spline Ti
mass        * 47.90
...
print       "Calculated cohesive energy: ${cohesive_energy} eV/atom"
print       "Reference cohesive energy:  -4.831 eV/atom"
... \\
13
-20.0 0.0
1 0 1 0
1.74269283716363254 6e+00 3.74427717596638934 9e+00 0.0e+00
2.05580176739999664 9e+00 9.10839730905501299 1e-01 1.08702523301094053 9e+01
2.36891069763636075 2e+00 3.88045896634145071 1e-01 -1.55322418835956632 7e+00
2.68201962787272441 0e+00 -1.88409065334698942 7e-02 2.43630041371478522 9e+00
...
```

Value perceptibility constraint

Chaos Control

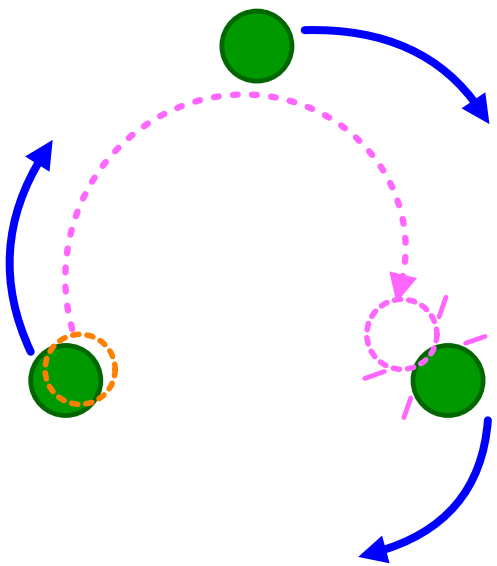
- What is vulnerable to input manipulations?

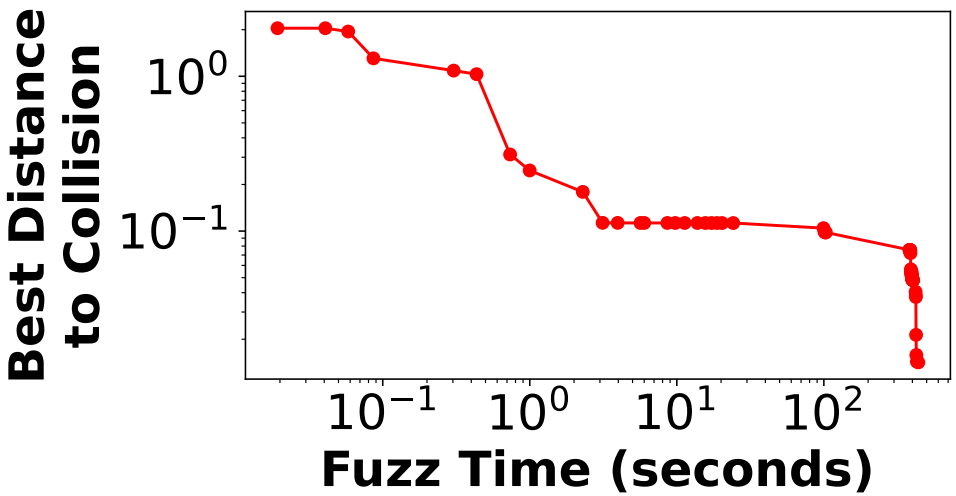
Chaos Control

- What is vulnerable to input manipulations?
- Chaotic dynamical systems
  - Vulnerability rooted in physics
  - Small change in initial state $\Rightarrow$ large change in final state
  - Proof-of-concept for nudging attacks

3-Body Simulation

Can we nudge the planets onto a collision trajectory? Yes!

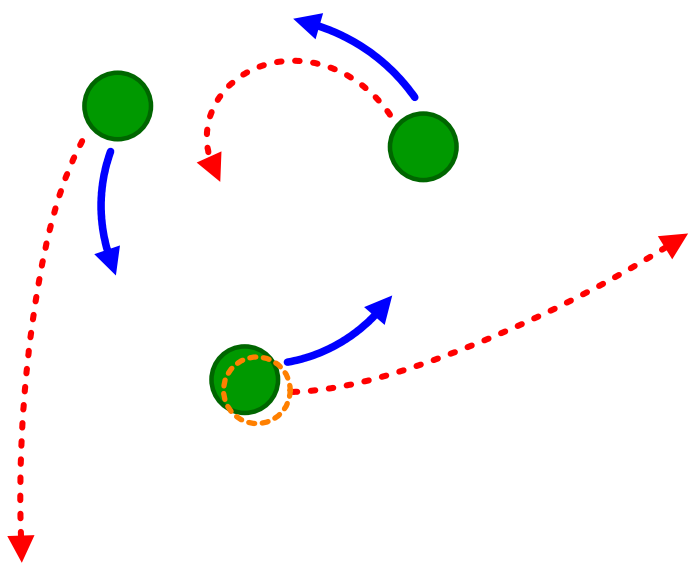


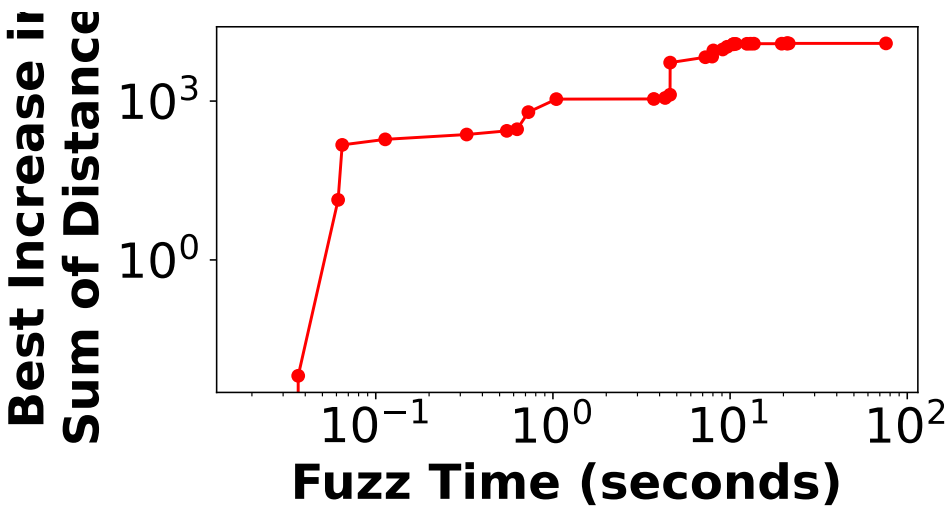


Distance to target (collision)

3-Body Simulation

Can we nudge the planets onto a divergence trajectory? Yes!





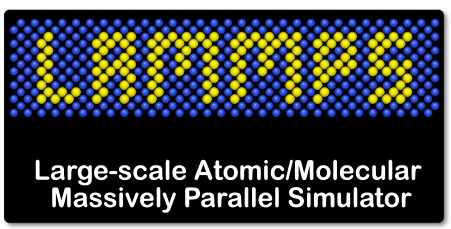
Distance from truth

(*increase in* sum of distances)

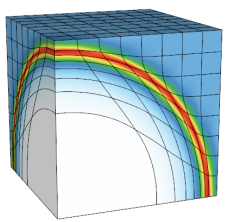
Larger Targets



GROMACS



LAMMPS



Laghos

Together: 2M+ lines of code

Nudging Attacks on Large Applications

LAMMPS: `in.alpha`

```
print       "Calculated cohesive energy: ${cohesive_energy} eV/atom"
print       "Reference cohesive energy:  -4.831 eV/atom"
... \\
1.74269283716363254 6e+00 3.74427717596638934 9e+00 0.0e+00
2.05580176739999664 9e+00 9.10839730905501299 1e-01 1.08702523301094053 9e+01
2.36891069763636075 2e+00 3.88045896634145071 1e-01 -1.55322418835956632 7e+00
2.68201962787272441 0e+00 -1.88409065334698942 7e-02 2.43630041371478522 9e+00
...
```

Nudging Attacks on Large Applications

LAMMPS: `in.alpha`

```
print       "Calculated cohesive energy: ${cohesive_energy} eV/atom"
print       "Reference cohesive energy:  -4.831 eV/atom"
... \\
1.7426928371636325 46e+00 3.7442771759663893 49e+00 0.0e+00
2.0558017673999966 49e+00 9.1083973090550129 91e-01 1.0870252330109405 39e+01
2.3689106976363607 52e+00 3.8804589663414507 11e-01 -1.5532241883595663 27e+00
2.6820196278727244 10e+00 -1.8840906533469894 27e-02 2.4363004137147852 29e+00
...
```

Nudging Attacks on Large Applications

LAMMPS: `in.alpha`

```
print       "Calculated cohesive energy: ${cohesive_energy} eV/atom"
print       "Reference cohesive energy:  -4.831 eV/atom"
... \\
1.7426928371636325 46e+00 3.7442771759663893 49e+00 0.0e+00
2.0558017673999966 49e+00 9.1083973090550129 91e-01 1.0870252330109405 39e+01
2.3689106976363607 52e+00 3.8804589663414507 11e-01 -1.5532241883595663 27e+00
2.6820196278727244 10e+00 -1.8840906533469894 27e-02 2.4363004137147852 29e+00
...
```

Metric value: $M = |\text{Nudged Output} - \text{Original Output}|$

Deflection ratio: $R = \left| \frac{|\text{Nudged Output} - \text{Original Output}|}{\text{Original Output}} \right|$

Nudging Attacks on Large Applications

LAMMPS: `in.alpha`

```
print       "Calculated cohesive energy: ${cohesive_energy} eV/atom"
print       "Reference cohesive energy:  -4.831 eV/atom"
... \\
1.7426928371636325 46e+00 3.7442771759663893 49e+00 0.0e+00
2.0558017673999966 49e+00 9.1083973090550129 91e-01 1.0870252330109405 39e+01
2.3689106976363607 52e+00 3.8804589663414507 11e-01 -1.5532241883595663 27e+00
2.6820196278727244 10e+00 -1.8840906533469894 27e-02 2.4363004137147852 29e+00
...
```

Metric value: $M = |\text{cohesive_energy} - \text{Original Output}|$

Deflection ratio: $R = \left| \frac{|\text{cohesive_energy} - \text{Original Output}|}{\text{Original Output}} \right|$

Nudging Attacks on Large Applications

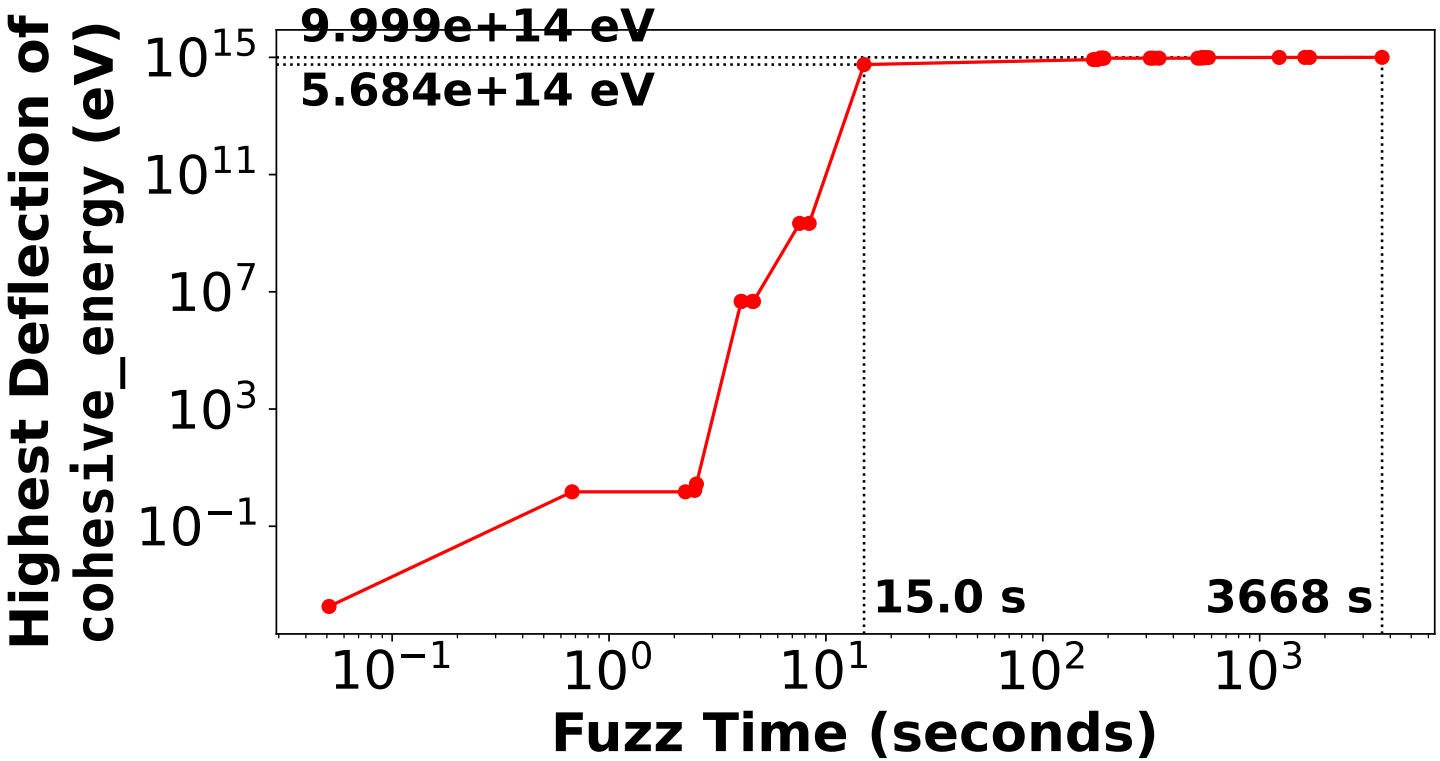
LAMMPS: `in.alpha`

```
print       "Calculated cohesive energy: ${cohesive_energy} eV/atom"
print       "Reference cohesive energy:  -4.831 eV/atom"
... \\
1.7426928371636325 46e+00 3.7442771759663893 49e+00 0.0e+00
2.0558017673999966 49e+00 9.1083973090550129 91e-01 1.0870252330109405 39e+01
2.3689106976363607 52e+00 3.8804589663414507 11e-01 -1.5532241883595663 27e+00
2.6820196278727244 10e+00 -1.8840906533469894 27e-02 2.4363004137147852 29e+00
...
```

Metric value: $M = |\text{cohesive_energy} - (-4.831)|$

Deflection ratio: $R = \left| \frac{|\text{cohesive_energy} - (-4.831)|}{(-4.831)} \right|$

How Quickly Can Deflection Occur?

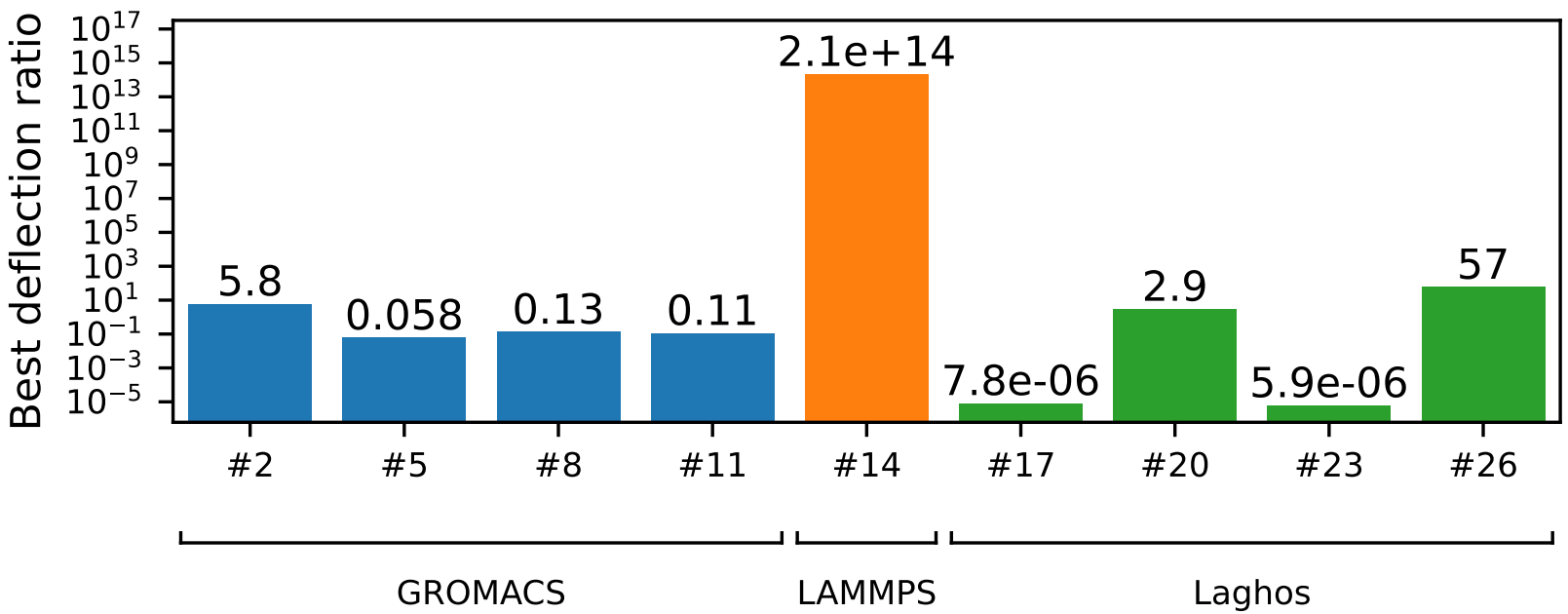


LAMMPS

`in.alpha`

Maximum Deflection Ratio

2 Hours, *Value* Perceptibility Constraint (Nudge $\leq 10^{-6}$)



Many have a notable deflection ratio

Other Perceptibility Constraints

Three quantifiers for **nudge perceptibility**

Other Perceptibility Constraints

Three quantifiers for **nudge perceptibility**

- $\mathbf{Value}(\boldsymbol{\Delta})$: v becomes a random value in $(v - \Delta, v + \Delta)$

Other Perceptibility Constraints

Three quantifiers for **nudge perceptibility**

- $\mathbf{Value}(\mathbf{10^{-6}})$: v becomes a random value in $(v - 10^{-6}, v + 10^{-6})$

Other Perceptibility Constraints

```
print       "Calculated cohesive energy: ${cohesive_energy} eV/atom"
print       "Reference cohesive energy:  -4.831 eV/atom"
... \\
1.74269283716363254 6e+00 3.74427717596638934 9e+00 0.0e+00
2.05580176739999664 9e+00 9.10839730905501299 1e-01 1.08702523301094053 9e+01
2.36891069763636075 2e+00 3.88045896634145071 1e-01 -1.55322418835956632 7e+00
2.68201962787272441 0e+00 -1.88409065334698942 7e-02 2.43630041371478522 9e+00
...
```

Other Perceptibility Constraints

```
print       "Calculated cohesive energy: ${cohesive_energy} eV/atom"
print       "Reference cohesive energy:  -4.831 eV/atom"
... \\
1.74269283716363632546e+00 3.744277175966389349e+00 0.0e+00
2.055801767399996649e+00 9.108397309055012991e-01 1.0870252330109405539e+01
2.368910697636360752e+00 3.880458966341450711e-01 -1.5532241883595663 27e+00
2.682019627872724410e+00 -1.884090653346989427e-02 2.436300413714785229e+00
...
```

Other Perceptibility Constraints

```
print       "Calculated cohesive energy: ${cohesive_energy} eV/atom"
print       "Reference cohesive energy:  -4.831 eV/atom"
... \\
1.74269283295363254 6e+00 3.7442771759663893 49e+00 0.0e+00
2.0558017673999966 49e+00 9.1083973090550129 91e-01 1.0870252330109861 39e+01
2.3689106976363607 52e+00 3.8804589663414507 11e-01 -1.5532241883595663 27e+00
2.6820196278727244 10e+00 -1.8840906533032894 27e-02 2.4363004137147852 29e+00
...
```

Other Perceptibility Constraints

Three quantifiers for **nudge perceptibility**

- $\mathbf{Value}(10^{-6})$: v becomes a random value in $(v - 10^{-6}, v + 10^{-6})$
- $\mathbf{Limited\text{-}Value}(\boldsymbol{\Delta})$: $\mathrm{Value}(\Delta)$ on *any single value*

Other Perceptibility Constraints

Three quantifiers for **nudge perceptibility**

- $\mathbf{Value}(10^{-6})$: v becomes a random value in $(v - 10^{-6}, v + 10^{-6})$
- $\mathbf{Limited\text{-}Value}(\mathbf{10^{-6}})$: $\mathrm{Value}(10^{-6})$ on *any single value*

Other Perceptibility Constraints

```
print       "Calculated cohesive energy: ${cohesive_energy} eV/atom"
print       "Reference cohesive energy:  -4.831 eV/atom"
... \\
1.74269283716363254 6e+00 3.74427717596638934 9e+00 0.0e+00
2.05580176739999664 9e+00 9.10839730905501299 1e-01 1.08702523301094053 9e+01
2.36891069763636075 2e+00 3.88045896634145071 1e-01 -1.55322418835956632 7e+00
2.68201962787272441 0e+00 -1.88409065334698942 7e-02 2.43630041371478522 9e+00
...
```

Other Perceptibility Constraints

```
print       "Calculated cohesive energy: ${cohesive_energy} eV/atom"
print       "Reference cohesive energy:  -4.831 eV/atom"
... \\
1.74269283716363254 6e+00 3.74427717596638934 9e+00 0.0e+00
2.05580176739999664 9e+00 9.10839730905501299 1e-01 1.08702523301094053 9e+01
2.36891069763636075 2e+00 3.88045896634145071 1e-01 -1.55322418835956632 7e+00
2.68201962787272441 0e+00 -1.88409065334698942 7e-02 2.43630041371478522 9e+00
...
```

Other Perceptibility Constraints

```
print       "Calculated cohesive energy: ${cohesive_energy} eV/atom"
print       "Reference cohesive energy:  -4.831 eV/atom"
... \\
1.74269283716363254 6e+00 3.74427717596638934 9e+00 0.0e+00
2.05580176739999664 9e+00 9.10839730905501299 1e-01 1.08702523301094053 9e+01
2.36891069763636075 2e+00 3.88045896634728071 1e-01 -1.55322418835956632 7e+00
2.68201962787272441 0e+00 -1.88409065334698942 7e-02 2.43630041371478522 9e+00
...
```

Other Perceptibility Constraints

Three quantifiers for **nudge perceptibility**

- $\mathbf{Value}(10^{-6})$: v becomes a random value in $(v - 10^{-6}, v + 10^{-6})$
- $\mathbf{Limited\text{-}Value}(10^{-6})$: $\mathrm{Value}(10^{-6})$ on *any single value*
- $\mathbf{Levenshtein}(n, d)$: n values edited such that the Levenshtein distance to each new value is below d

Other Perceptibility Constraints

Three quantifiers for **nudge perceptibility**

- $\mathbf{Value}(10^{-6})$: v becomes a random value in $(v - 10^{-6}, v + 10^{-6})$
- $\mathbf{Limited\text{-}Value}(10^{-6})$: $\mathrm{Value}(10^{-6})$ on *any single value*
- $\mathbf{Levenshtein}(2, 1)$: 2 values edited such that the Levenshtein distance to each new value is below 1

Other Perceptibility Constraints

```
print       "Calculated cohesive energy: ${cohesive_energy} eV/atom"
print       "Reference cohesive energy:  -4.831 eV/atom"
... \\
1.7426928371636325 46e+00 3.7442771759663893 49e+00 0.0e+00
2.0558017673999966 49e+00 9.1083973090550129 91e-01 1.0870252330109405 39e+01
2.3689106976363607 52e+00 3.8804589663414507 11e-01 -1.5532241883595663 27e+00
2.6820196278727244 10e+00 -1.8840906533469894 27e-02 2.4363004137147852 29e+00
...
```

Other Perceptibility Constraints

```
print       "Calculated cohesive energy: ${cohesive_energy} eV/atom"
print       "Reference cohesive energy:  -4.831 eV/atom"
... \\
1.7426928371636325 46e+00 3.74427717596638934 9e+00 0.0e+00
2.0558017673999966 49e+00 9.108397309055012991e-01 1.0870252330109405 39e+01
2.3689106976363607 52e+00 3.8804589663414507 11e-01 -1.5532241883595663 27e+00
2.6820196278727244 10e+00 -1.8840906533469894 27e-02 2.436300413714785229e+00
...
```

Other Perceptibility Constraints

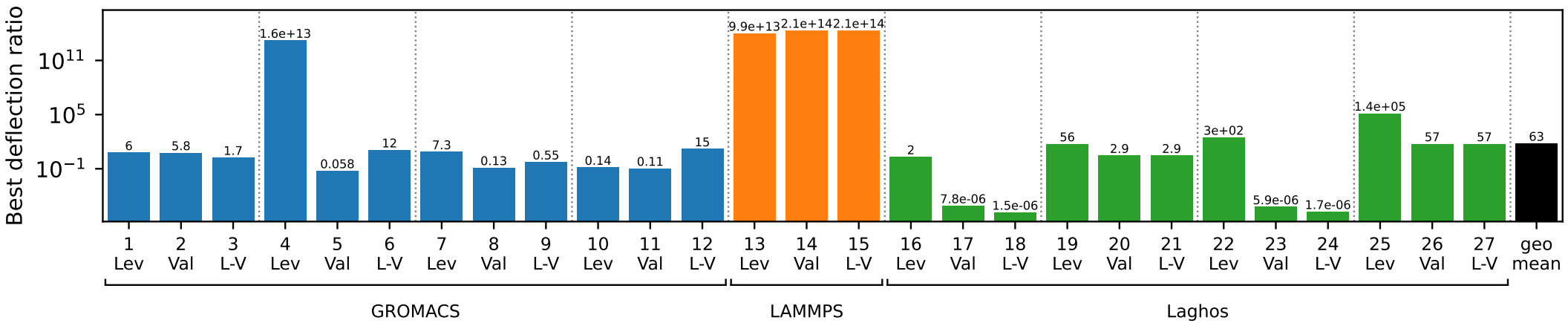
```
print       "Calculated cohesive energy: ${cohesive_energy} eV/atom"
print       "Reference cohesive energy:  -4.831 eV/atom"
... \\
1.7426928371636325 46e+00 3.74427717596638934 9e+00 0.0e+00
2.0558017673999966 49e+00 91.083973090550129 91e-01 1.0870252330109405 39e+01
2.3689106976363607 52e+00 3.8804589663414507 11e-01 -1.5532241883595663 27e+00
2.6820196278727244 10e+00 -1.8840906533469894 27e-02 2.4736300413714785 229e+00
...
```

Other Perceptibility Constraints

Three quantifiers for **nudge perceptibility**

- $\mathbf{Value}(\mathbf{10^{-6}})$: v becomes a random value in $(v - 10^{-6}, v + 10^{-6})$
- $\mathbf{Limited\text{-}Value}(\mathbf{10^{-6}})$: $\mathrm{Value}(10^{-6})$ on *any single value*
- $\mathbf{Levenshtein}(\mathbf{2}, \mathbf{1})$: 2 values edited such that the Levenshtein distance to each new value is below 1

All Experiments



Deflection ratio: $R = \left| \frac{\text{Nudged Output} - \text{Original Output}}{\text{Original Output}} \right|$

(*Note the logarithmic scale*)

Future Work

- Steering the output of large applications

Future Work

- Steering the output of large applications
- Preserving the structure of the computational problem

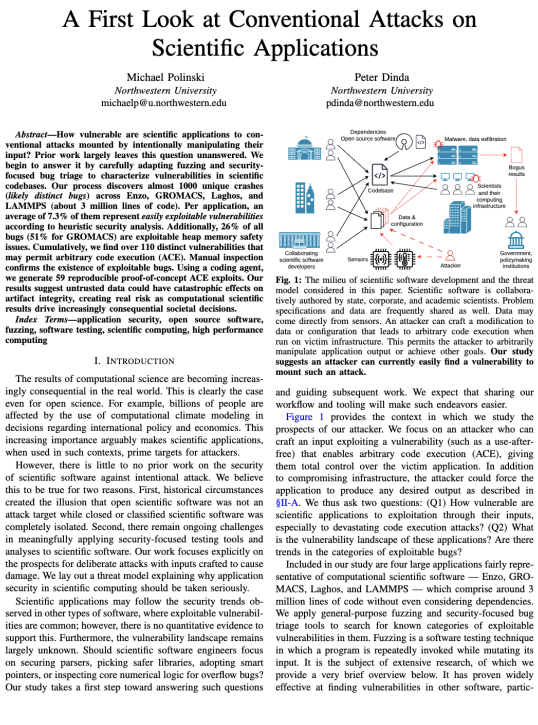
Future Work

- Steering the output of large applications
- Preserving the structure of the computational problem
- Automation with AI agents

Coming Soon

Scientific software has conventional vulnerabilities

- Upcoming presentation at SC'26 **S-HPC Workshop** on security
- **Fuzzing** finds bugs exploitable to achieve **arbitrary code execution**



A First Look at Conventional Attacks on Scientific Applications

Michael Polinski
Northwestern University
michaelp@u.northwestern.edu

Peter Dinda
Northwestern University
pdinda@northwestern.edu

***Abstract*—How vulnerable are scientific applications to conventional attacks mounted by intentionally manipulating their input? Prior work largely leaves this question unanswered. We begin to answer it by carefully adapting fuzzing and security-focused bug triage to characterize vulnerabilities in scientific codebases. Our process discovers almost 1000 unique crashes (*likely distinct bugs*) across Enzo, GROMACS, Laghos, and LAMMPS (about 3 million lines of code). Per application, an average of 7.3% of them represent *easily exploitable vulnerabilities* according to heuristic security analysis. Additionally, 26% of all bugs (51% for GROMACS) are exploitable heap memory safety issues. Cumulatively, we find over 110 distinct vulnerabilities that may permit arbitrary code execution (ACE). Manual inspection confirms the existence of exploitable bugs. Using a coding agent, we generate 59 reproducible proof-of-concept ACE exploits. Our results suggest untrusted data could have catastrophic effects on artifact integrity, creating real risk as computational scientific results drive increasingly consequential societal decisions.**

***Index Terms*—application security, open source software, fuzzing, software testing, scientific computing, high performance computing**

I. Introduction

The results of computational science are becoming increasingly consequential in the real world. This is clearly the case even for open science. For example, billions of people are affected by the use of computational climate modeling in decisions regarding international policy and economics. This increasing importance arguably makes scientific applications, when used in such contexts, prime targets for attackers.

However, there is little to no prior work on the security of scientific software against intentional attack. We believe this to be true for two reasons. First, historical circumstances created the illusion that open scientific software was not an attack target while closed or classified scientific software was completely isolated. Second, there remain ongoing challenges in meaningfully applying security-focused testing tools and analyses to scientific software. Our work focuses explicitly on the prospects for deliberate attacks with inputs crafted to cause damage. We lay out a threat model explaining why application security in scientific computing should be taken seriously.

Scientific applications may follow the security trends observed in other types of software, where exploitable vulnerabilities are common; however, there is no quantitative evidence to support this. Furthermore, the vulnerability landscape remains largely unknown. Should scientific software engineers focus on securing parsers, picking safer libraries, adopting smart pointers, or inspecting core numerical logic for overflow bugs? Our study takes a first step toward answering such questions and guiding subsequent work. We expect that sharing our workflow and tooling will make such endeavors easier.

SC26 Workshops, November 15-20, 2026, Chicago, Illinois, USA
979-8-3195-1221-5/26/$31.00 ©2026 IEEE

Fig. 1: The milieu of scientific software development and the threat model considered in this paper. Scientific software is collaboratively authored by state, corporate, and academic scientists. Problem specifications and data are frequently shared as well. Data may come directly from sensors. An attacker can craft a modification to data or configuration that leads to arbitrary code execution when run on victim infrastructure. This permits the attacker to arbitrarily manipulate application output or achieve other goals. **Our study suggests an attacker can currently easily find a vulnerability to mount such an attack.**

Figure 1 provides the context in which we study the prospects of our attacker. We focus on an attacker who can craft an input exploiting a vulnerability (such as a use-after-free) that enables arbitrary code execution (ACE), giving them total control over the victim application. In addition to compromising infrastructure, the attacker could force the application to produce any desired output as described in §II-A. We thus ask two questions: (Q1) How vulnerable are scientific applications to exploitation through their inputs, especially to devastating code execution attacks? (Q2) What is the vulnerability landscape of these applications? Are there trends in the categories of exploitable bugs?

Included in our study are four large applications fairly representative of computational scientific software — Enzo, GROMACS, Laghos, and LAMMPS — which comprise around 3 million lines of code without even considering dependencies. We apply general-purpose fuzzing and security-focused bug triage tools to search for known categories of exploitable vulnerabilities in them. Fuzzing is a software testing technique in which a program is repeatedly invoked while mutating its input. It is the subject of extensive research, of which we provide a very brief overview below. It has proven widely effective at finding vulnerabilities in other software, partic-

Thank You!

Michael Polinski

https://michaelpolinski.com

Buoyancy Project

https://buoyancy-project.org

Prescience Lab

http://presciencelab.org

Northwestern University