Automated Software Testing with Blackbox Fuzzing

Vaggelis Atlidakis

University of Southern California,
Nov 2023, Invited Talk

BROWN

Why software testing?

- U.S. companies lost > $2 trillion in 2022 due to poor software quality[1]
  - ~10% of U.S. Gross Domestic Product (GDP)

- Bugs found earlier are easier to fix
  - A bug found in coding/unit testing takes ~3x less time to fix than in post-release[2]

This talk: Automated software testing with blackbox fuzzing

- RESTler [ICSE '19]: Collaboration with Microsoft research
- IvySyn [USENIX SEC '23]: Collaboration with Brown University

[1] "Cost of Poor Software Quality: A 2022 Report," Consortium for Information & Software Quality

[2] "Impact of Inadequate Infrastructure for Software Testing, 2022," National Institute of Standards & Technology

Outline

Why software testing?

- U.S. companies lost > $2 trillion in 2022 due to poor software quality[1]
  - ~10% of U.S. Gross Domestic Product (GDP)
- Bugs found earlier are easier to fix
  - A bug found in coding/unit testing takes ~3x less time to fix than in post-release[2]

This talk: Automated software testing with blackbox fuzzing

- RESTler [ICSE '19]: Collaboration with Microsoft research
- IvySyn [USENIX SEC '23]: Collaboration with Brown University

[1] "Cost of Poor Software Quality: A 2022 Report," Consortium for Information & Software Quality

[2] "Impact of Inadequate Infrastructure for Software Testing, 2022," National Institute of Standards & Technology

RESTler: Stateful REST API Fuzzing

What

- Test cloud services with REST APIs
- Find ``500 Internal Server Errors''

Why

- Past approaches were not automated
- Testing one API request each time

How: *Stateful REST API Fuzzing*

- Generate *stateful* sequences of API requests

Test example on GitLab[1]

Delete a file from a project

1. Create a gitlab project
2. Create a file
3. Delete the file

➢ “500 Internal Server Error”

producer of: project-id

```
curl --request POST --header "PRIVATE-TOKEN: <your-token>" \
     --header "Content-Type: application/json" --data '{
        "name": "new_project", "description": "New Project", "path": "new_project",
        "namespace_id": "42", "initialize_with_readme": "true"}' \
     --url "https://gitlab.example.com/api/v4/projects/"
```

project-id: 13083

consumer of: project-id

producer of: file-name

```
curl --request POST --header 'PRIVATE-TOKEN: <your_access_token>' \
     --header "Content-Type: application/json" \
     --data '{"branch": "main", "author_email": "author@example.com", "author_name": "Firstname Lastname",
              "content": "some content", "commit_message": "create a new file"}' \
     "https://gitlab.example.com/api/v4/projects/13083/repository/files/app%2Fproject%2Erb"
```

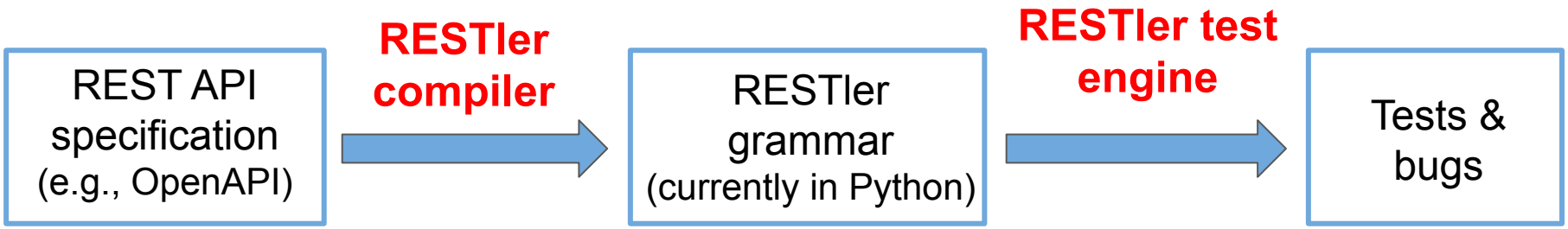
consumer of: project-id, file-name

file-name: app/project.rb

```
curl --request DELETE --header 'PRIVATE-TOKEN: <your_access_token>' \
     --header "Content-Type: application/json" \
     --data '{"branch": "main", "author_email": "author@example.com", "author_name": "Firstname Lastname",
       "commit_message": "          "}' \
     "https://gitlab.example.com/api/v4/projects/13083/repository/files/app%2Fproject%2Erb"
```

[1] https://about.gitlab.com/

System overview



- Identify producer-consumer relationships
- Generate code to parse responses

- Generate tests (state-space exploration)
- Drop invalid requests

Selected evaluation results

- Q1: Do RESTler tests help increase code coverage?
- Q2: Bugs found with RESTler?

Study subject: Gitlab

- Complex backend (> 350 KLOCs; mostly ruby-on-rails)
- Hundrends of API endpoints
- Complex API request payloads

Deeper service exploration (Q1)

API Family	Total API Requests	Seq. Len.	Code Coverage (lines of code)	Tests
Gitlab	15 (*11)	1	598	1
Commits		2	1108	7
		3	1196	250
		4	1760	2220
		…	…	…

Testing APIs with RESTler (5h per API family)

- Longer sequences increase service-side code coverage

Deeper service exploration (Q1)

API Family	Total API Requests	Seq. Len.	Code Coverage (lines of code)	Tests
Gitlab	15 (*11)	1	598	1
Commits		2	1108	7
		3	1196	250
		4	1760	2220
		...	...	...

Testing APIs with RESTler (5h per API family)

- Longer sequences increase service-side code coverage
- Progress in large search space

Deeper service exploration (Q1)

API Family	Total API Requests	Seq. Len.	Code Coverage (lines of code)	Tests
Gitlab	15 (*11)	1	598	1
Commits		2	1108	7
		3	1196	250
		4	1760	2220
		...	...	...

Testing APIs with RESTler (5h per API family)

- ❖ Longer sequences increase service-side code coverage
- ❖ Progress in large search space

Example: Testing for 5 hours

- ➢ **Brute-force**: 741 million sequences
- ➢ **RESTler:** 2220 total test sequences
  - ✓ Producer-consumer request dependencies
  - ✓ Dynamic service feedback

Bugs found with RESTler (Q2)

Gitlab Bug [#50268]

1. Create a gitlab project
2. Create a file with a proper commit message
3. Delete the file with an empty commit message

➢ "500 Internal Server Error"

➢ Found 28 confirmed such bugs in Gitlab in 2018

➢ …and many more in production cloud services

RESTler since 2019

- Uncovered 100s of bugs in production Azure, Bing, and Office365 services
  - Including "*severe critical bugs*[1]"
- Open-sourced at: https://github.com/microsoft/restler-fuzzer
- Multiple teams are using it daily
  - Developer fuzzing
  - Specification correctness
  - Regression testing

[1] https://patricegodefroid.github.io/research-overview.html

Outline

Why software testing?

- U.S. companies lost > $2 trillion in 2022 due to poor software quality[1]
  - ~10% of U.S. Gross Domestic Product (GDP)
- Bugs found earlier are easier to fix
  - A bug found in coding/unit testing takes ~3x less time to fix than in post-release[2]

This talk: Automated software testing with blackbox fuzzing

- RESTler [ICSE '19]: Collaboration with Microsoft research
- IvySyn [USENIX SEC '23]: Collaboration with Brown University

[1] "Cost of Poor Software Quality: A 2022 Report," Consortium for Information & Software Quality

[2] "Impact of Inadequate Infrastructure for Software Testing, 2022," National Institute of Standards & Technology

IvySyn: Automated Vulnerability Discovery in DL Frameworks

What

- Find memory safety errors in Deep Learning (DL) frameworks

Why

- Core of DL frameworks is implemented in C/C++
- Past approaches required manual effort and domain knowledge

How: A bottom-up approach

- Find a crash in the native part
- Synthesize python code reproducing the crash

Example

Native (*kernel*) implementation

```
class EditDistanceOp : public OpKernel {
  void Compute(OpKernelContext* ctx) override {
  // ...
  if (g_truth == g_hypothesis) {
  auto loc = std::inner_product(g_truth.begin(),
     g_truth.end(), output_strides.begin(),
     int64_t{0});
  OP_REQUIRES(
      loc < output_elements,
      errors::Internal("..."))
  output_t(loc) =
      gtl::LevenshteinDistance<T>(truth_seq,
        hypothesis_seq, cmp);
  // ...
```

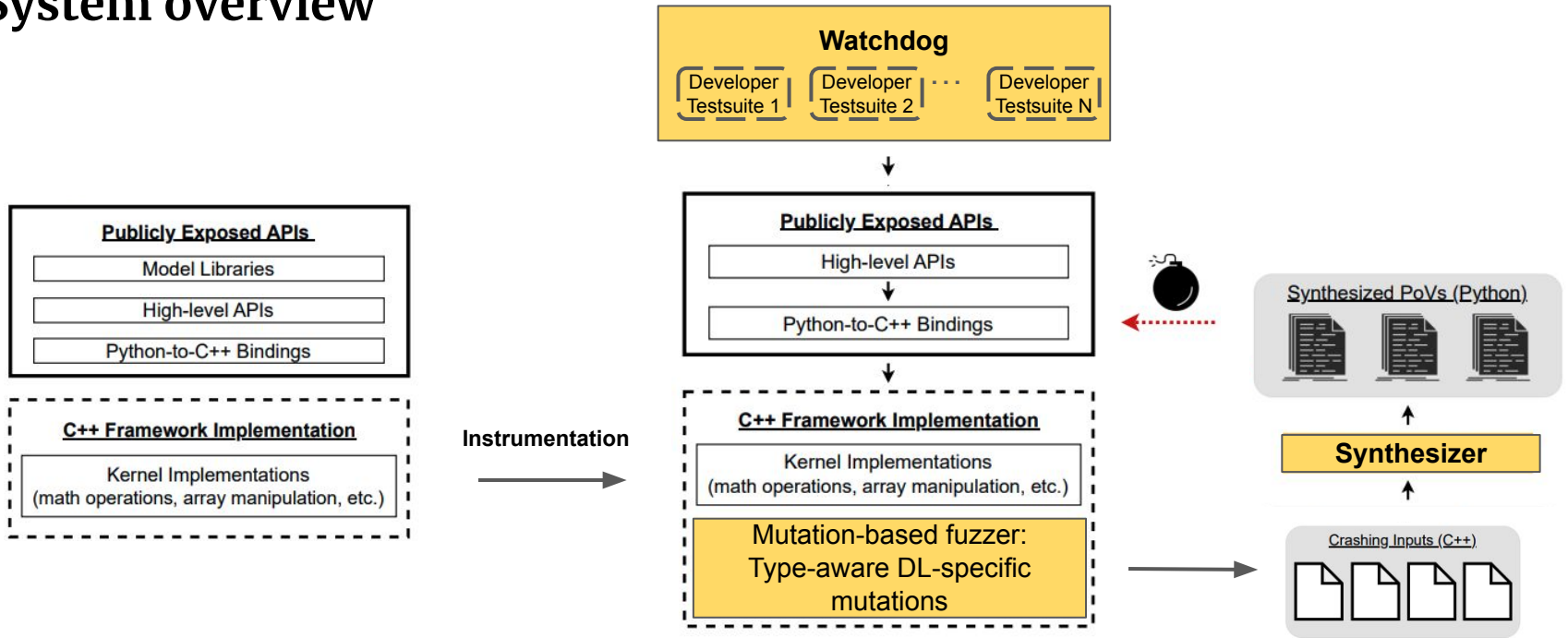
Triggers a crash

from public python APIs

Proof-of-Vulnerability (PoV)

```
hypothesis_indices = tf.constant(-125099896764,
                     shape=[3,3], dtype=tf.int64)
hypothesis_values  = tf.constant(0,
                     shape=[3], dtype=tf.int64)
hypothesis_shape   = tf.constant(0,
                     shape=[3], dtype=tf.int64)
truth_indices      = tf.constant(0,
                     shape=[2,3], dtype=tf.int64)
truth_values       = tf.constant(-1879048192,
                     shape=[2], dtype=tf.int64)
truth_shape        = tf.constant(2,
                     shape=[3], dtype=tf.int64)
tf.raw_ops.EditDistance(
    hypothesis_indices = hypothesis_indices,
    hypothesis_values  = hypothesis_values,
    hypothesis_shape   = hypothesis_shape,
    truth_indices      = truth_indices,
    truth_values       = truth_values,
    truth_shape        = truth_shape)
```

System overview



Selected evaluation results

- Q1: How quickly can IvySyn find crashes?
- Q2: How effective is IvySyn in synthesizing PoVs?

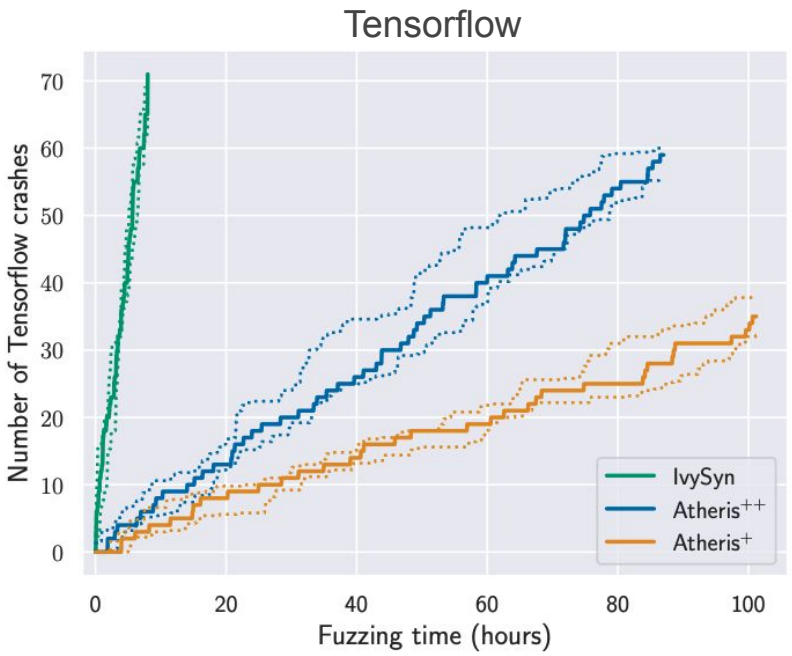
Target Frameworks: *Tensorflow*[1] and *Pytorch*[2]

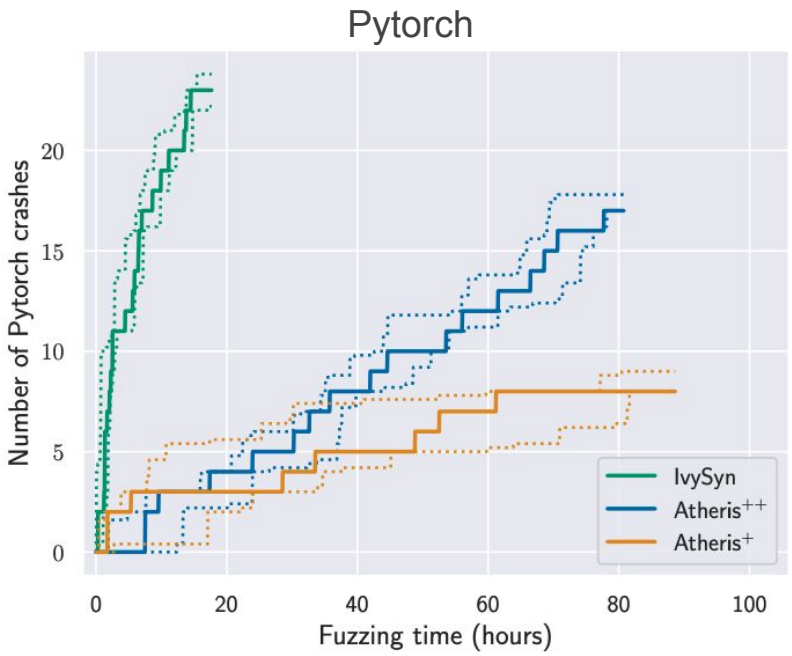
Against Google's *Atheris*[3]

- ***Atheris+***: Default *Atheris* + automation
- **Atheris++**: Default *Atheris* + automation + type-aware
- **IvySyn:** Type-aware + DL-specific mutations

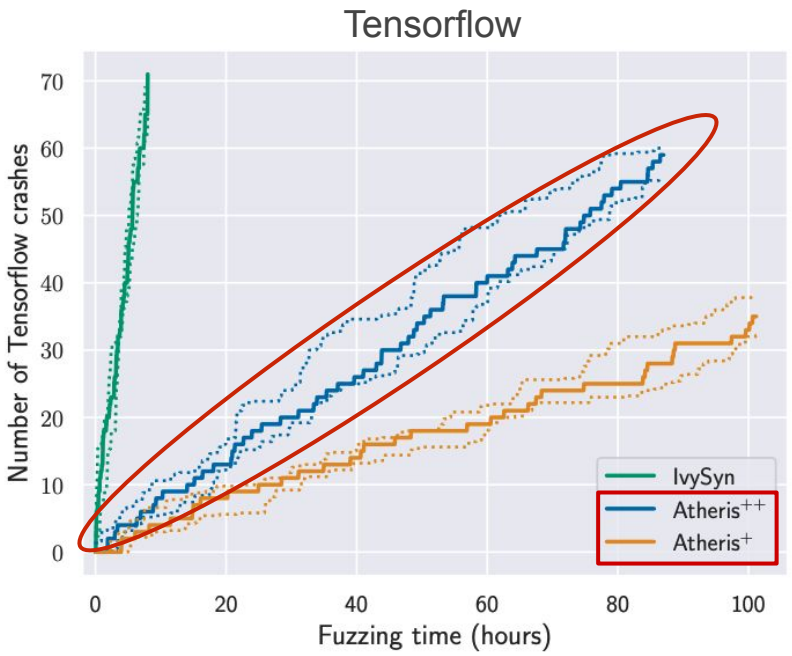
[1] https://www.tensorflow.org/
[2] https://github.com/pytorch/pytorch
[3] https://github.com/google/atheris

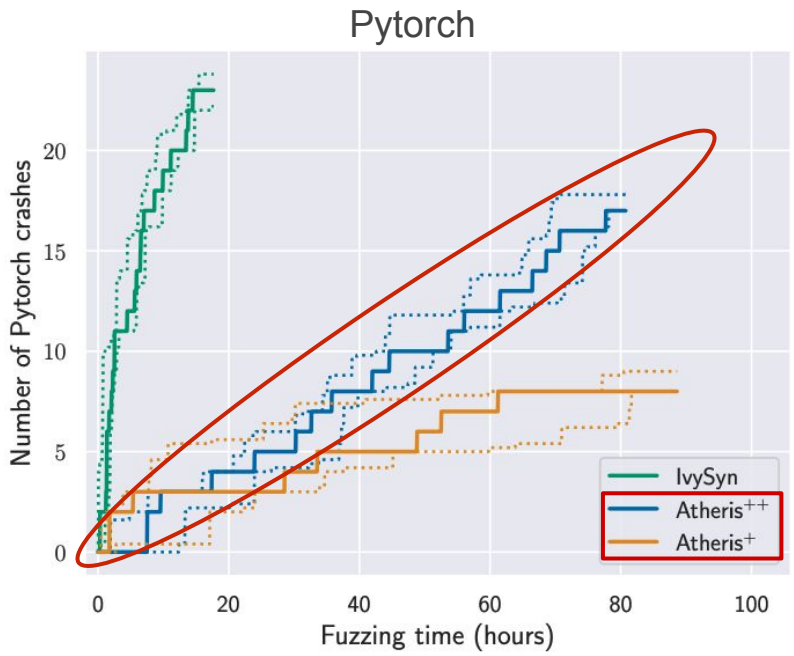
Efficiency in finding crashing inputs (Q1)



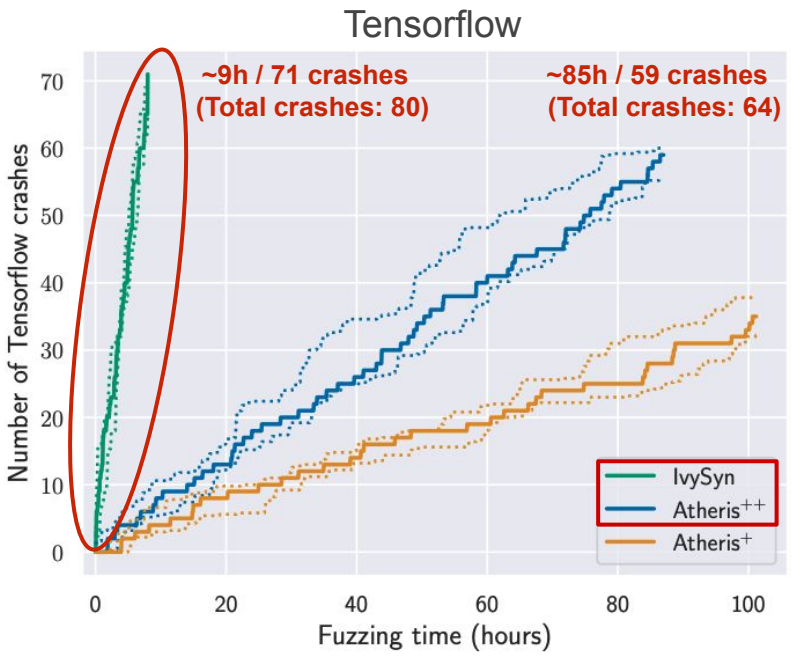


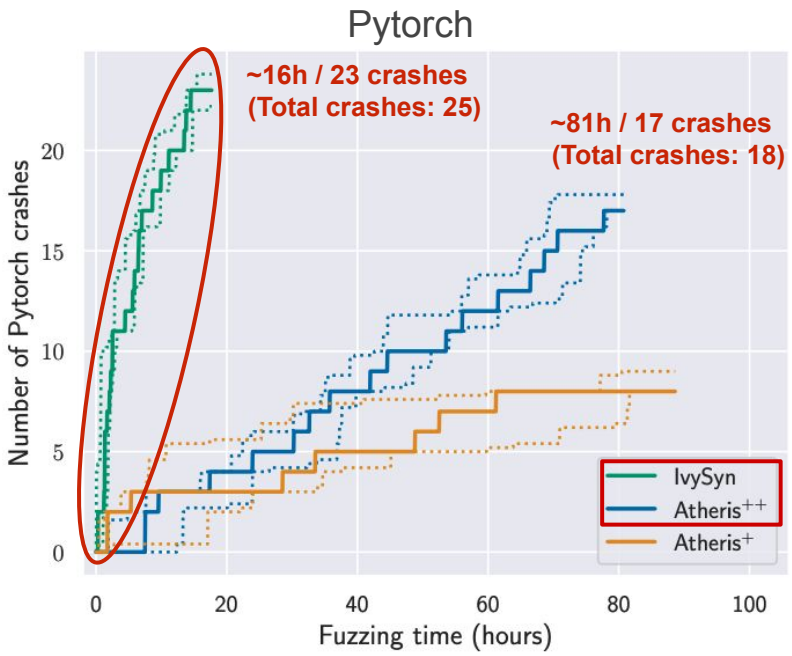
Efficiency in finding crashing inputs (Q1)



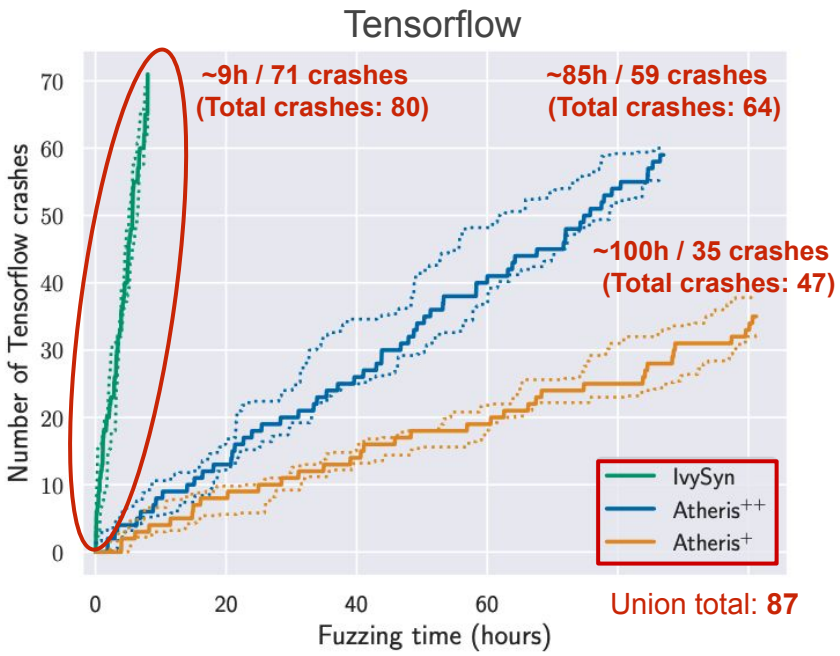


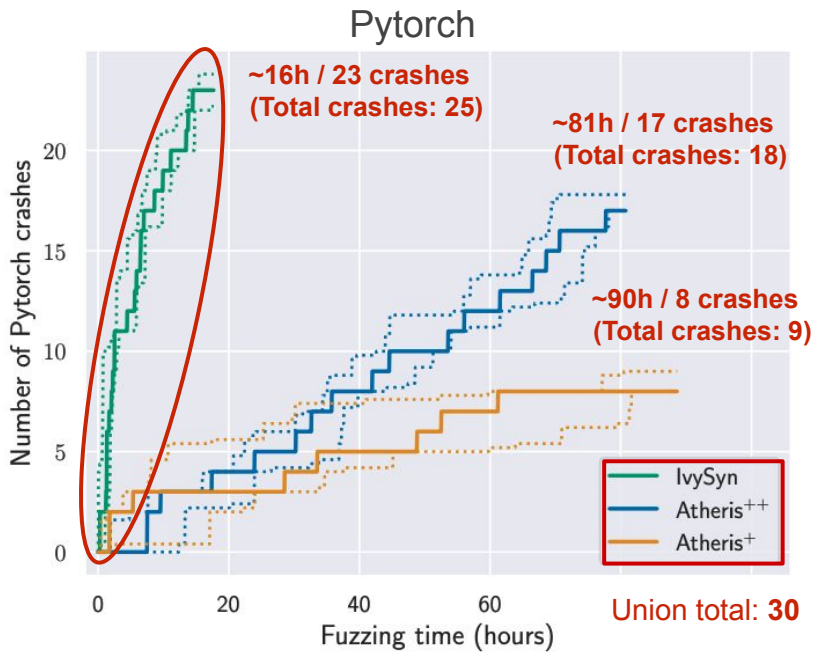
Efficiency in finding crashing inputs (Q1)





Efficiency in finding crashing inputs (Q1)





Effectiveness in synthesizing PoVs (Q2)

Framework	Fuzzed Kernels	Unique Crashes	Synthesized PoVs
Tensorflow	412	103	86 / 103 (83%)
Pytorch	747	81	49 / 81 (60%)
All	1159	184	135 / 184 (73%)

- Synthesized 135 PoVs
- Identified 61 previously-unknown vulnerabilities
- Assigned with **39** new CVEs

https://gitlab.com/brown-ssl/ivysyn

Effectiveness in synthesizing PoVs (Q2)

Framework	Fuzzed Kernels	Unique Crashes	Synthesized PoVs
Tensorflow	412	103	86 / 103 (83%)
Pytorch	747	81	49 / 81 (60%)
All	1159	184	135 / 184 (73%)

- Synthesized 135 PoVs
- Identified 61 previously-unknown vulnerabilities
- Assigned with **39** new CVEs

https://gitlab.com/brown-ssl/ivysyn

```
import tensorflow as tf
hypothesis_indices = tf.constant(-125099896764,
                                 shape=[3,3], dtype=tf.int64)
hypothesis_values  = tf.constant(0,
                                 shape=[3], dtype=tf.int64)
hypothesis_shape   = tf.constant(0,
                                 shape=[3], dtype=tf.int64)
truth_indices      = tf.constant(0,
                                 shape=[2,3], dtype=tf.int64)
truth_values       = tf.constant(-1879048192,
                                 shape=[2], dtype=tf.int64)
truth_shape        = tf.constant(2,
                                 shape=[3], dtype=tf.int64)

tf.raw_ops.EditDistance(
    hypothesis_indices = hypothesis_indices,
    hypothesis_values  = hypothesis_values,
    hypothesis_shape   = hypothesis_shape,
    truth_indices      = truth_indices,
    truth_values       = truth_values,
    truth_shape        = truth_shape)
```

```
class EditDistanceOp : public OpKernel {
  void Compute(OpKernelContext* ctx) override {
  // ...
  if (g_truth == g_hypothesis) {
  auto loc = std::inner_product(g_truth.begin(),
     g_truth.end(), output_strides.begin(),
     int64_t{0});
  OP_REQUIRES(
      ctx,
      0 <= loc &&
      loc < output_elements,
      errors::Internal("...")
  output_t(loc) =
      gtl::LevenshteinDistance<T>(truth_seq,
        hypothesis_seq, cmp);
  // ...
```

Automated software testing with blackbox fuzzing

- RESTler [ICSE '19]
  Top-down approach: Start from the APIs
- IvySyn [USENIX SEC '23]
  Bottom-up approach: Start from native implementations

RESTler team: Patrice Godefroid and Marina Polishchuk @ Microft Research

IvySyn team: Neophytos Chrisou, Di Jin, and Vasilions Kemerlis @ Brown University
Baishakhi Ray @ Columbia University