IvySyn

Automated Vulnerability Discovery in Deep Learning Frameworks

Neophytos Christou[1] Di Jin[1] Vaggelis Atlidakis[1] Baishakhi Ray[2]
Vasileios P. Kemerlis[1]

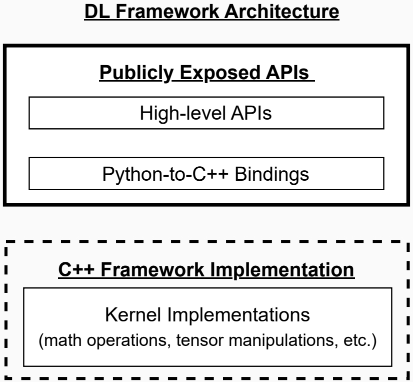
August 9, 2023

[1]Secure Systems Laboratory (SSL)
Department of Computer Science
Brown University

[2]ARiSE Lab
Department of Computer Science
Columbia University

BROWN

Motivation

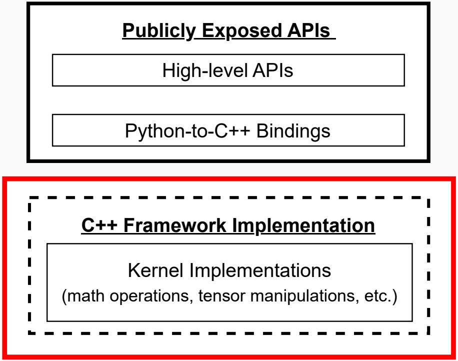


*https://github.com/tensorflow/tensorflow/blob/master/SECURITY.md

Motivation

- Core DL framework implementation ➜ memory-unsafe languages

DL Framework Architecture



*https://github.com/tensorflow/tensorflow/blob/master/SECURITY.md

Motivation

- Core DL framework implementation ➜ memory-unsafe languages
- 200+ memory-safety related CVEs in 2021-2022 alone (Tensorflow) *

TFSA-2021-126	Use after free in boosted trees creation
TFSA-2021-125	Heap buffer overflow in `FractionalAvgPoolGrad`
TFSA-2021-124	Segfault and heap buffer overflow in `{Experimental,}DatasetToTFRecord`
TFSA-2021-123	Null pointer dereference in `UncompressElement`
TFSA-2021-122	Incorrect validation of `SaveV2` inputs
TFSA-2021-121	Null pointer dereference in `SparseTensorSliceDataset`
TFSA-2021-120	Bad alloc in `StringNGrams` caused by integer conversion

*https://github.com/tensorflow/tensorflow/blob/master/SECURITY.md

Goals and Past Approaches

◎ IvySyn's Goals

Goals and Past Approaches

IvySyn's Goals

- Automatically uncover *memory safety* and *fatal runtime* errors in DL frameworks

Goals and Past Approaches

IvySyn's Goals

- Automatically uncover *memory safety* and *fatal runtime* errors in DL frameworks
- Help framework developers *identify* and *fix* the uncovered bugs

BROWN

Goals and Past Approaches

IvySyn's Goals

- Automatically uncover *memory safety* and *fatal runtime* errors in DL frameworks
- Help framework developers *identify* and *fix* the uncovered bugs

Past Approaches

Goals and Past Approaches

IvySyn's Goals

- Automatically uncover *memory safety* and *fatal runtime* errors in DL frameworks
- Help framework developers *identify* and *fix* the uncovered bugs

Past Approaches

- Are not aimed at finding memory safety errors

BROWN

Goals and Past Approaches

IvySyn's Goals

- Automatically uncover *memory safety* and *fatal runtime* errors in DL frameworks
- Help framework developers *identify* and *fix* the uncovered bugs

Past Approaches

- Are not aimed at finding memory safety errors
- Are not *fully automated*
  - Custom fuzzing drivers, domain-expert annotations, ...

BROWN

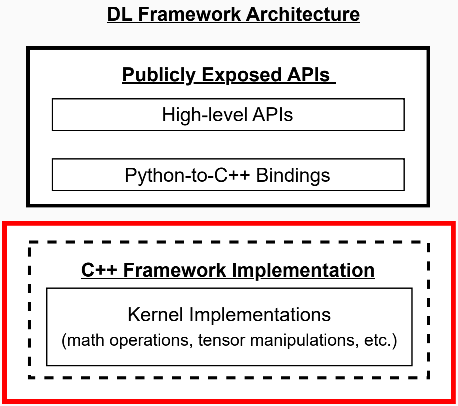
IvySyn Overview

➥ IvySyn's Approach

IvySyn Overview

➥ IvySyn's Approach

- Fuzz the native implementation of DL frameworks



IvySyn Overview

➥ IvySyn's Approach

- Fuzz the native implementation of DL frameworks
- Automatically synthesize *Proof-of-Vulnerability (PoV)* snippets

PoV synthesized by IvySyn

```
import tensorflow as tf

indices = tf.constant([], shape=[2,0],
                   dtype=tf.int64)
values = tf.constant([2,0,1], shape=[3],
                   dtype=tf.int64)
dense_shape = tf.constant([2,0,1], shape=[3],
                   dtype=tf.int64)
default_value = tf.constant(0, shape=[],
                   dtype=tf.int64)
tf.raw_ops.SparseFillEmptyRows(
               indices=indices,
               values=values,
               dense_shape=dense_shape,
               default_value=default_value)
```

PoV triggers crash!

```
$ python3 pov.py
segmentation fault (core dumped)
```

BROWN

IvySyn Overview

IvySyn's Approach

- Fuzz the native implementation of DL frameworks
- Automatically synthesize *Proof-of-Vulnerability (PoV)* snippets

Achievements

PoV synthesized by IvySyn

```
import tensorflow as tf

indices = tf.constant([], shape=[2,0],
                      dtype=tf.int64)
values = tf.constant([2,0,1], shape=[3],
                      dtype=tf.int64)
dense_shape = tf.constant([2,0,1], shape=[3],
                      dtype=tf.int64)
default_value = tf.constant(0, shape=[],
                      dtype=tf.int64)
tf.raw_ops.SparseFillEmptyRows(
               indices=indices,
               values=values,
               dense_shape=dense_shape,
               default_value=default_value)
```

PoV triggers crash!

```
$ python3 pov.py
segmentation fault (core dumped)
```

BROWN

IvySyn Overview

➥ IvySyn's Approach

- Fuzz the native implementation of DL frameworks
- Automatically synthesize *Proof-of-Vulnerability (PoV)* snippets

Achievements

- Uncovered 61 previously-unknown vulnerabilities

PoV synthesized by IvySyn

```
import tensorflow as tf

indices = tf.constant([], shape=[2,0],
                   dtype=tf.int64)
values = tf.constant([2,0,1], shape=[3],
                   dtype=tf.int64)
dense_shape = tf.constant([2,0,1], shape=[3],
                   dtype=tf.int64)
default_value = tf.constant(0, shape=[],
                   dtype=tf.int64)
tf.raw_ops.SparseFillEmptyRows(
               indices=indices,
               values=values,
               dense_shape=dense_shape,
               default_value=default_value)
```

PoV triggers crash!

```
$ python3 pov.py
segmentation fault (core dumped)
```

IvySyn Overview

IvySyn's Approach

- Fuzz the native implementation of DL frameworks
- Automatically synthesize *Proof-of-Vulnerability (PoV)* snippets

Achievements

- Uncovered 61 previously-unknown vulnerabilities
- Assigned with 39 unique CVEs

PoV synthesized by IvySyn

```
import tensorflow as tf

indices = tf.constant([], shape=[2,0],
                      dtype=tf.int64)
values = tf.constant([2,0,1], shape=[3],
                      dtype=tf.int64)
dense_shape = tf.constant([2,0,1], shape=[3],
                      dtype=tf.int64)
default_value = tf.constant(0, shape=[],
                      dtype=tf.int64)
tf.raw_ops.SparseFillEmptyRows(
               indices=indices,
               values=values,
               dense_shape=dense_shape,
               default_value=default_value)
```

PoV triggers crash!

```
$ python3 pov.py
segmentation fault (core dumped)
```

BROWN

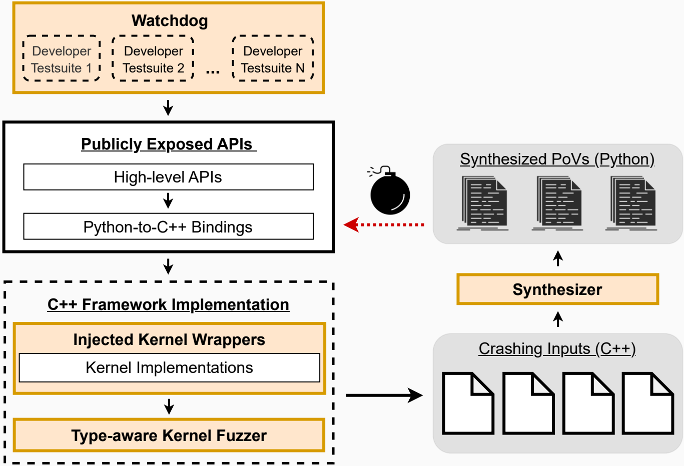
Outline

IvySyn Architecture

Evaluation

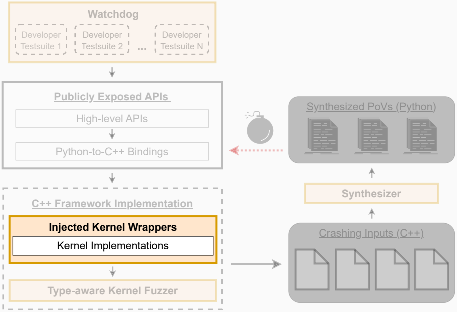
Conclusion

IvySyn Architecture Overview



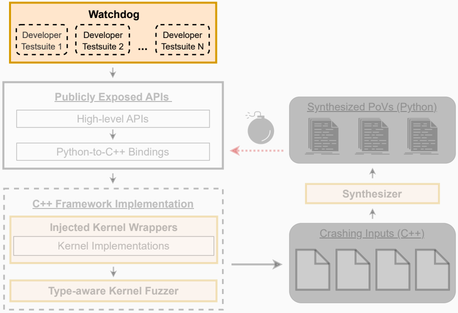
IvySyn Architecture ➜ Kernel Instrumentation

- `IvySyn` automatically wraps each framework's *kernels* with fuzzing drivers



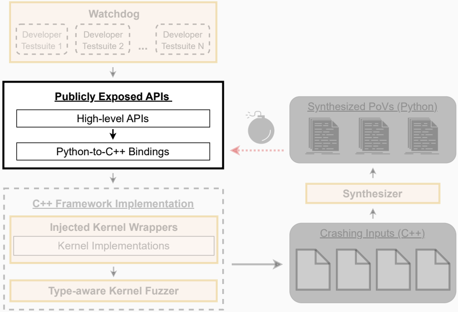
IvySyn Architecture ➜ Force-executing Instrumented Kernels

- `IvySyn` runs DL frameworks' developer-provided unit tests



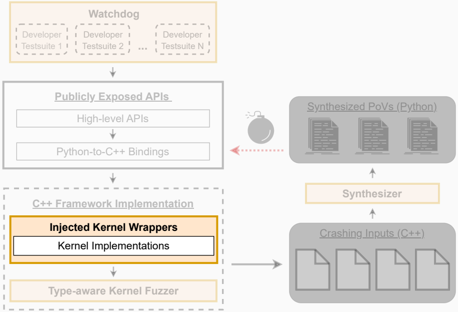
IvySyn Architecture ➜ Force-executing Instrumented Kernels

- `IvySyn` runs DL frameworks' developer-provided unit tests
- Execution reaches the *instrumented* kernels ...



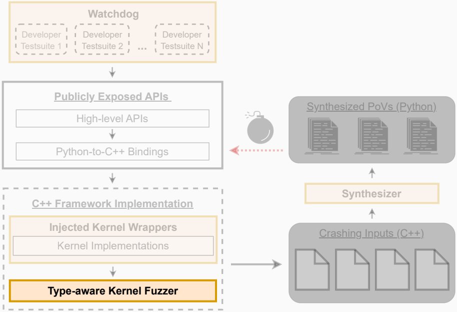
IvySyn Architecture ➜ Force-executing Instrumented Kernels

- `IvySyn` runs DL frameworks' developer-provided unit tests
- Execution reaches the *instrumented* kernels ...



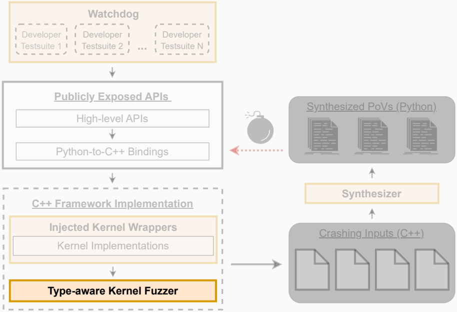
IvySyn Architecture ➜ Force-executing Instrumented Kernels

- `IvySyn` runs DL frameworks' developer-provided unit tests
- Execution reaches the *instrumented* kernels ...
- ...and bootstraps a fuzzing session



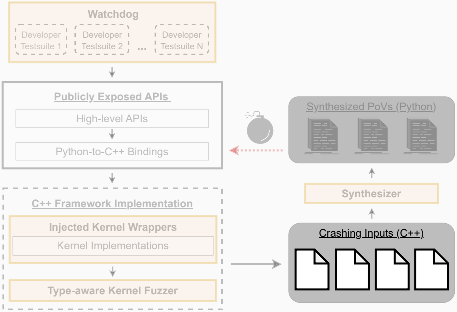
IvySyn Architecture ➜ Type-aware Fuzzer

- Performs type-aware mutations based on the original argument types



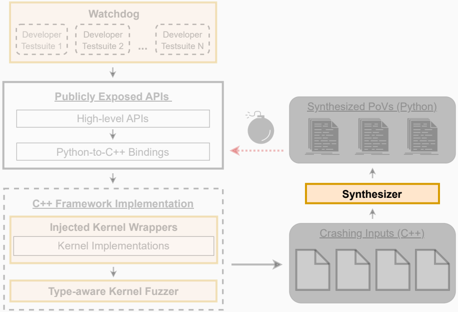
IvySyn Architecture ➜ Type-aware Fuzzer

- Performs type-aware mutations based on the original argument types
- Logs *native crashing inputs* in crash-reports



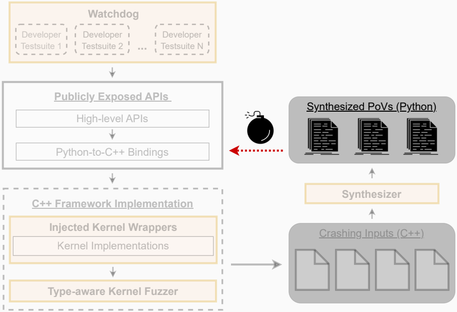
IvySyn Architecture ➜ PoV Synthesis

- Logged crash-reports are fed into `IvySyn`'s synthesizer

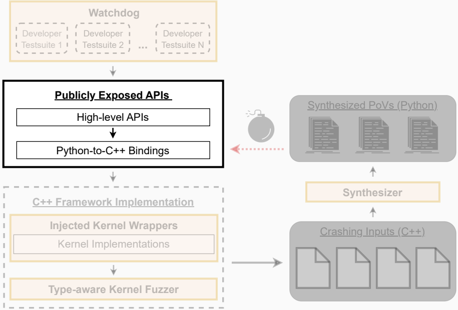


IvySyn Architecture ➜ PoV Synthesis

- Logged crash-reports are fed into `IvySyn`'s synthesizer
- The synthesizer generates *Proof-of-Vulnerability (PoV)* snippets



- Logged crash-reports are fed into `IvySyn`'s synthesizer
- The synthesizer generates *Proof-of-Vulnerability (PoV)* snippets
- The PoVs trigger the native crashes from publicly exposed Python APIs



IvySyn Architecture ➜ PoV Synthesis (cont'd)

Crash-report produced by IvySyn

```
# SparseFillEmptyRowsOp
Tensor<type: int64 shape: [2,0] values: >
Tensor<type: int64 shape: [3] values: 2 0 1>
Tensor<type: int64 shape: [3] values: 2 0 1>
Tensor<type: int64 shape: [] values: 0>
```

Crash-report produced by IvySyn

```
# SparseFillEmptyRowsOp
Tensor<type: int64 shape: [2,0] values: >
Tensor<type: int64 shape: [3] values: 2 0 1>
Tensor<type: int64 shape: [3] values: 2 0 1>
Tensor<type: int64 shape: [] values: 0>
```

</> Corresponding PoV synthesized by IvySyn

```python
import tensorflow as tf

indices = tf.constant([], shape=[2,0],
                   dtype=tf.int64)
values = tf.constant([2,0,1], shape=[3],
                   dtype=tf.int64)
dense_shape = tf.constant([2,0,1], shape=[3],
                   dtype=tf.int64)
default_value = tf.constant(0, shape=[],
                   dtype=tf.int64)
tf.raw_ops.SparseFillEmptyRows(
               indices=indices,
               values=values,
               dense_shape=dense_shape,
               default_value=default_value)
```

Crash-report produced by IvySyn

```
# SparseFillEmptyRowsOp
Tensor<type: int64 shape: [2,0] values: >
Tensor<type: int64 shape: [3] values: 2 0 1>
Tensor<type: int64 shape: [3] values: 2 0 1>
Tensor<type: int64 shape: [] values: 0>
```

PoV triggers crash!

```
$ python3 pov.py
segmentation fault (core dumped)
```

</> Corresponding PoV synthesized by IvySyn

```python
import tensorflow as tf

indices = tf.constant([], shape=[2,0],
                   dtype=tf.int64)
values = tf.constant([2,0,1], shape=[3],
                   dtype=tf.int64)
dense_shape = tf.constant([2,0,1], shape=[3],
                   dtype=tf.int64)
default_value = tf.constant(0, shape=[],
                   dtype=tf.int64)
tf.raw_ops.SparseFillEmptyRows(
               indices=indices,
               values=values,
               dense_shape=dense_shape,
               default_value=default_value)
```

Outline

IvySyn Architecture

Evaluation

Conclusion

Evaluation

TensorFlow

PyTorch

BROWN

Evaluation

TensorFlow PyTorch

Q1: How *efficient* is `IvySyn` at uncovering crashing inputs?

TensorFlow PyTorch

Q1: How *efficient* is `IvySyn` at uncovering crashing inputs?

Q2: How *effective* is `IvySyn` at leveraging crashing inputs to synthesize PoVs?

TensorFlow PyTorch

Q1: How *efficient* is `IvySyn` at uncovering crashing inputs?

Q2: How *effective* is `IvySyn` at leveraging crashing inputs to synthesize PoVs?

Q3: Which `IvySyn` mutations are the most successful in uncovering memory errors?

Q1: How *efficient* is IvySyn at uncovering crashing inputs?

- Compared IvySyn's *efficiency* at uncovering crashes against Atheris†

†*Atheris: A Coverage-Guided, Native Python Fuzzer.* Google.

Evaluation ➜ IvySyn vs Atheris

Q1: How *efficient* is IvySyn at uncovering crashing inputs?

- Compared IvySyn's *efficiency* at uncovering crashes against Atheris†
- Leveraged IvySyn's argument logging functionality and synthesizer to generate *fuzzing drivers* for Atheris

† *Atheris: A Coverage-Guided, Native Python Fuzzer.* Google.

Q1: How *efficient* is IvySyn at uncovering crashing inputs?

- Compared IvySyn's *efficiency* at uncovering crashes against Atheris†
- Leveraged IvySyn's argument logging functionality and synthesizer to generate *fuzzing drivers* for Atheris
- Generated two different variants of drivers

†*Atheris: A Coverage-Guided, Native Python Fuzzer.* Google.

Q1: How *efficient* is IvySyn at uncovering crashing inputs?

- Compared IvySyn's *efficiency* at uncovering crashes against Atheris[†]
- Leveraged IvySyn's argument logging functionality and synthesizer to generate *fuzzing drivers* for Atheris
- Generated two different variants of drivers

Atheris⁺

- Drivers without type awareness
- Atheris randomly chooses argument types

[†] *Atheris: A Coverage-Guided, Native Python Fuzzer.* Google.

Q1: How *efficient* is IvySyn at uncovering crashing inputs?

- Compared IvySyn's *efficiency* at uncovering crashes against Atheris[†]
- Leveraged IvySyn's argument logging functionality and synthesizer to generate *fuzzing drivers* for Atheris
- Generated two different variants of drivers

Atheris⁺

- Drivers without type awareness
- Atheris randomly chooses argument types

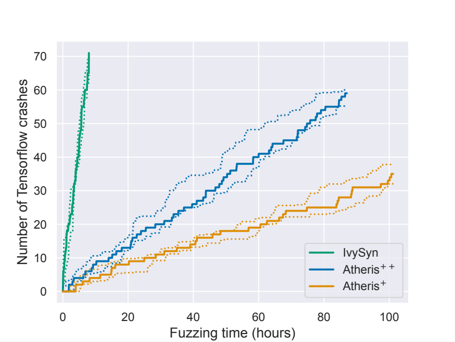
Atheris⁺⁺

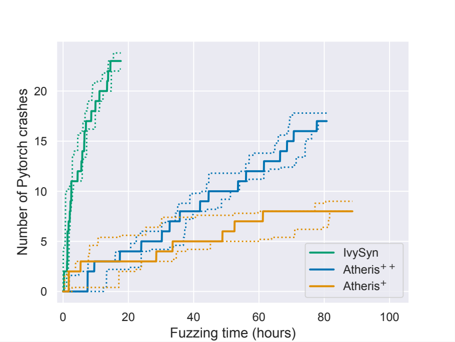
- Drivers with type awareness
- The drivers provide Atheris with the proper argument types

[†] *Atheris: A Coverage-Guided, Native Python Fuzzer.* Google.

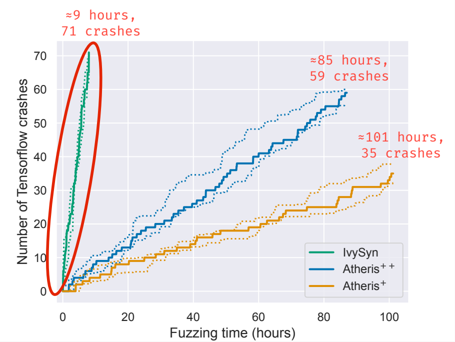
Evaluation → IvySyn vs Atheris (cont'd)

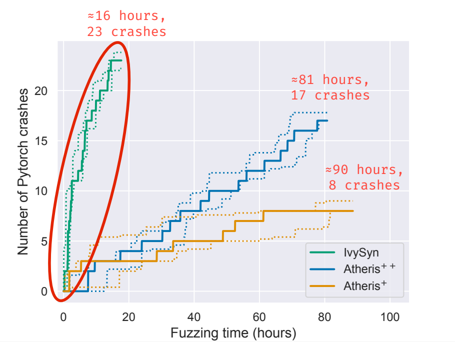
Q1: How *efficient* is IvySyn at uncovering crashing inputs?





Q1: How *efficient* is IvySyn at uncovering crashing inputs?





- IvySyn uncovers more crashes than Atheris, and does so faster

Q1: How *efficient* is IvySyn at uncovering crashing inputs?

Number of crashes found by IvySyn vs Atheris (over 5 iterations)

	Fuzzer	TensorFlow	PyTorch
Total Crashes	Atheris+	47	9
	Atheris++	64	18
	IvySyn	**80**	**25**
Union	All	87	30

- IvySyn uncovers more crashes than Atheris, and does so faster

Evaluation → IvySyn vs DocTer

Q2: How *effective* is IvySyn at leveraging crashing inputs to synthesize PoVs?

- Compared IvySyn's *effectiveness* at synthesizing PoVs against the *semi-automated* DocTer[‡] tool

[‡] *DocTer: Documentation-Guided Fuzzing for Testing Deep Learning API Functions.* Xie *et al.*

Evaluation ➜ IvySyn vs DocTer

Q2: How *effective* is IvySyn at leveraging crashing inputs to synthesize PoVs?

- Compared IvySyn's *effectiveness* at synthesizing PoVs against the *semi-automated* DocTer‡ tool

Synthesized PoVs (IvySyn vs DocTer) (over 10 iterations)

Fuzzer	Total		Median		Median Running Time (mins)	
	TensorFlow	PyTorch	TensorFlow	PyTorch	TensorFlow	PyTorch
DocTer	16	9	12	7	199	736
IvySyn	19	14	15	11	184	569

- IvySyn synthesizes more PoVs than DocTer, *without* manual effort

‡ *DocTer: Documentation-Guided Fuzzing for Testing Deep Learning API Functions.* Xie *et al.*

BROWN

Q2: How *effective* is IvySyn at leveraging crashing inputs to synthesize PoVs?

Accumulated IvySyn results

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%)**

Q2: How *effective* is `IvySyn` at leveraging crashing inputs to synthesize PoVs?

Accumulated IvySyn results

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%)**

- `IvySyn` synthesized 135 PoVs and was attributed with 39 CVEs

BROWN

Q3: Which `IvySyn` mutations are the most successful in uncovering memory errors?

Number of PoVs per mutation type

`IvySyn` Type-aware Mutation Type	Total PoVs
Tensors with random dimension sizes	46
Tensors with extreme values	25
Permutations of original arguments	19
Zero values	12
Lists with extreme values	13
Tensors with empty shape	8
Extreme values in primitive types	6
Empty lists	3
Deep tensors	3

Q3: Which `IvySyn` mutations are the most successful in uncovering memory errors?

Number of PoVs per mutation type

`IvySyn` Type-aware Mutation Type	Total PoVs
Tensors with random dimension sizes	46
Tensors with extreme values	25
Permutations of original arguments	19
Zero values	12
Lists with extreme values	13
Tensors with empty shape	8
Extreme values in primitive types	6
Empty lists	3
Deep tensors	3

- DL-specific (*e.g.*, tensor) mutations are especially effective

BROWN

Outline

IvySyn Architecture

Evaluation

Conclusion

Conclusion

- Fully-automated framework

Conclusion

- Fully-automated framework
  - Perform type-aware, DL-specific mutations

BROWN

Conclusion

- Fully-automated framework
  - Perform type-aware, DL-specific mutations
  - Fuzz *native implementation* of DL frameworks

Conclusion

- Fully-automated framework
  - Perform type-aware, DL-specific mutations
  - Fuzz *native implementation* of DL frameworks
  - Synthesize PoVs to trigger detected crashes from Python

BROWN

Conclusion

- Fully-automated framework
  - Perform type-aware, DL-specific mutations
  - Fuzz *native implementation* of DL frameworks
  - Synthesize PoVs to trigger detected crashes from Python
- Identified 61 previously-unknown vulnerabilities

BROWN

Conclusion

- Fully-automated framework
  - Perform type-aware, DL-specific mutations
  - Fuzz *native implementation* of DL frameworks
  - Synthesize PoVs to trigger detected crashes from Python
- Identified 61 previously-unknown vulnerabilities
- Assigned with 39 unique CVEs

Conclusion

- Fully-automated framework
  - Perform type-aware, DL-specific mutations
  - Fuzz *native implementation* of DL frameworks
  - Synthesize PoVs to trigger detected crashes from Python
- Identified 61 previously-unknown vulnerabilities
- Assigned with 39 unique CVEs

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

BROWN