IvySyn
Automated Vulnerability Discovery in Deep Learning Frameworks

Neophytos Christou\textsuperscript{1} Di Jin\textsuperscript{1} Vaggelis Atlidakis\textsuperscript{1} Baishakhi Ray\textsuperscript{2}
Vasileios P. Kemerlis\textsuperscript{1}
August 9, 2023

\textsuperscript{1}Secure Systems Laboratory (SSL)
Department of Computer Science
Brown University

\textsuperscript{2}ARiSE Lab
Department of Computer Science
Columbia University

neophytos_christou@brown.edu (Brown University)
Motivation

**DL Framework Architecture**

- **Publicly Exposed APIs**
  - High-level APIs
  - Python-to-C++ Bindings

- **C++ Framework Implementation**
  - Kernel Implementations
    (math operations, tensor manipulations, etc.)

---

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

neophytos_christou@brown.edu (Brown University)  IvySyn

USENIX Security 2023  1 / 19
Motivation

- Core DL framework implementation → memory-unsafe languages

DL Framework Architecture

- Publicly Exposed APIs
  - High-level APIs
  - Python-to-C++ Bindings

- C++ Framework Implementation
  - Kernel Implementations
    (math operations, tensor manipulations, etc.)

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

neophytos_christou@brown.edu (Brown University)  IvySyn
Motivation

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

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

neophytos_christou@brown.edu (Brown University)  IvySyn

USENIX Security 2023
Goals and Past Approaches

🎯 IvySyn’s Goals

- **Automatically** uncover *memory safety* and *fatal runtime* errors in DL frameworks
IvySyn’s Goals

- Automatically uncover memory safety and fatal runtime errors in DL frameworks
- Help framework developers identify and fix the uncovered bugs
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, …
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
## Goals and Past Approaches

<table>
<thead>
<tr>
<th>IvySyn’s Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <strong>Automatically</strong> uncover <em>memory safety</em> and <em>fatal runtime</em> errors in DL frameworks</td>
</tr>
<tr>
<td>- Help framework developers <em>identify</em> and <em>fix</em> the uncovered bugs</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Past Approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Are <strong>not</strong> aimed at finding memory safety errors</td>
</tr>
<tr>
<td>- Are <strong>not</strong> fully automated</td>
</tr>
<tr>
<td>- Custom fuzzing drivers, domain-expert annotations, ...</td>
</tr>
</tbody>
</table>
IvySyn Overview

- IvySyn’s Approach

Achievements

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

IvySyn’s Approach

- Fuzz the native implementation of DL frameworks

DL Framework Architecture

Publicly Exposed APIs

- High-level APIs
- Python-to-C++ Bindings

C++ Framework Implementation

- Kernel Implementations
  (math operations, tensor manipulations, etc.)
IvySyn Overview

➡️ IvySyn’s Approach

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

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)
```

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

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)
```

PoV triggers crash!

$ python3 pov.py
segmentation fault (core dumped)

neophytos_christou@brown.edu (Brown University)

IvySyn

USENIX Security 2023
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

```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)
```

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

```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)
```

PoV triggers crash!

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

neophytos_christou@brown.edu (Brown University)
IvySyn Architecture

Evaluation

Conclusion
IvySyn Architecture Overview

- **Watchdog**
  - Developer Testsuite 1
  - Developer Testsuite 2
  - ... Developer Testsuite N

- **Publicly Exposed APIs**
  - High-level APIs
  - Python-to-C++ Bindings

- **C++ Framework Implementation**
  - Injected Kernel Wrappers
    - Kernel Implementations
  - Type-aware Kernel Fuzzer

- **Synthesizer**
  - Synthesized PoVs (Python)

- **Crashing Inputs (C++)**

- Developer

- Testsuite 1
- Testsuite 2
- ... Testsuite N

**Type-aware Kernel Fuzzer**

**C++ Framework Implementation**

**Publicly Exposed APIs**

**Watchdog**

**IvySyn**

Brown University
**IvySyn automatically wraps each framework’s kernels with fuzzing drivers**

- **IvySyn Architecture ➔ Kernel Instrumentation**

  - **Developer**
  - Testsuite 1
  - Testsuite N

  - **Watchdog**
  - **Publicly Exposed APIs**
    - High-level APIs
    - Python-to-C++ Bindings

  - **C++ Framework Implementation**
    - **Injected Kernel Wrappers**
      - Kernel Implementations
    - Type-aware Kernel Fuzzer

  - **Synthesized PoVs (Python)**
  - **Crashing Inputs (C++)**
  - **Synthesizer**

  - neophytos_christou@brown.edu (Brown University)

  - IvySyn

  - Brown University

  - USENIX Security 2023
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 ...

---

Diagram elements:
- Publicly Exposed APIs
  - High-level APIs
  - Python-to-C++ Bindings
- C++ Framework Implementation
  - Injected Kernel Wrappers
  - Kernel Implementations
  - Type-aware Kernel Fuzzer
- Synthesizer
- Crashing Inputs (C++)
- Synthesized PoVs (Python)
- Watchdog
  - Developer
  - Testsuite 1
  - Testsuite 2
  - ... 
  - Testsuite N

---

neophytos_christou@brown.edu (Brown University)  IvySyn

USENIX Security 2023  7 / 19
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
Logged crash-reports are fed into IvySyn's synthesizer.
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**.
Crash-report produced by IvySyn

```python
# 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)
```
IvySyn Architecture ➔ PoV Synthesis (cont’d)

Crash-report produced by IvySyn

```python
# 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)
```

neophytos_christou@brown.edu (Brown University) IvySyn
Crash-report produced by IvySyn

```python
# 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!

```bash
$ python3 pov.py
```

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)
```

neophytos_christou@brown.edu (Brown University)
Outline

IvySyn Architecture

Evaluation

Conclusion
Evaluation

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?
Evaluation

Q1: How efficient is IvySyn at uncovering crashing inputs?
Q1: How efficient is IvySyn at uncovering crashing inputs?
Q2: How effective is IvySyn at leveraging crashing inputs to synthesize PoVs?
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?
Evaluation ➾ IvySyn vs Atheris

Q1: How efficient is IvySyn at uncovering crashing inputs?

- Compared IvySyn’s efficiency at uncovering crashes against 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**

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
- Generated two different variants of drivers


neophytos_christou@brown.edu (Brown University)    IvySyn  USENIX Security 2023  13 / 19
**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**

- Generated two different variants of drivers

---

**Atheris**

- Drivers **without** type awareness

- **Atheris** randomly chooses argument types

---

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

neophytos_christou@brown.edu (Brown University)

IvySyn

USENIX Security 2023 13 / 19
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
- Generated two different variants of drivers

<table>
<thead>
<tr>
<th>Atheris†</th>
<th>Atheris++</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Drivers without type awareness</td>
<td>- Drivers with type awareness</td>
</tr>
<tr>
<td>- Atheris randomly chooses argument types</td>
<td>- The drivers provide Atheris with the proper argument types</td>
</tr>
</tbody>
</table>

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

**Graphs:**
- **IvySyn:** 
  - Fuzzing time: ≈9 hours, 71 crashes
  - Fuzzing time: ≈85 hours, 59 crashes
  - Fuzzing time: ≈101 hours, 35 crashes

- **Atheris:** 
  - Fuzzing time: ≈16 hours, 23 crashes
  - Fuzzing time: ≈81 hours, 17 crashes
  - Fuzzing time: ≈90 hours, 8 crashes

**Observation:**
- **IvySyn** uncovers **more crashes** than **Atheris**, and does so **faster**
**Q1:** How *efficient* is IvySyn at uncovering crashing inputs?

<table>
<thead>
<tr>
<th>Total Crashes</th>
<th>Fuzzer</th>
<th>TensorFlow</th>
<th>PyTorch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atheris⁺</td>
<td>47</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Atheris⁺⁺</td>
<td>64</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>IvySyn</td>
<td>80</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>Union</td>
<td>All</td>
<td>87</td>
<td>30</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Fuzzer</th>
<th>Total TensorFlow</th>
<th>Median TensorFlow</th>
<th>Median Running Time (mins) TensorFlow</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocTer</td>
<td>16</td>
<td>12</td>
<td>199</td>
</tr>
<tr>
<td>IvySyn</td>
<td>19</td>
<td>15</td>
<td>184</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fuzzer</th>
<th>Total PyTorch</th>
<th>Median PyTorch</th>
<th>Median Running Time (mins) PyTorch</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocTer</td>
<td>9</td>
<td>7</td>
<td>736</td>
</tr>
<tr>
<td>IvySyn</td>
<td>14</td>
<td>11</td>
<td>569</td>
</tr>
</tbody>
</table>

- IvySyn synthesizes more PoVs than DocTer, without manual effort

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

eophytos_christou@brown.edu (Brown University)
Q2: How effective is IvySyn at leveraging crashing inputs to synthesize PoVs?

<table>
<thead>
<tr>
<th>Framework</th>
<th>Fuzzed Kernels</th>
<th>Unique Crashes</th>
<th>Synthesized PoVs</th>
</tr>
</thead>
<tbody>
<tr>
<td>TensorFlow</td>
<td>412</td>
<td>103</td>
<td>86 / 103 (83%)</td>
</tr>
<tr>
<td>PyTorch</td>
<td>747</td>
<td>81</td>
<td>49 / 81 (60%)</td>
</tr>
<tr>
<td>All</td>
<td>1159</td>
<td>184</td>
<td>135 / 184 (73%)</td>
</tr>
</tbody>
</table>
Q2: How effective is IvySyn at leveraging crashing inputs to synthesize PoVs?

<table>
<thead>
<tr>
<th>Framework</th>
<th>Fuzzed Kernels</th>
<th>Unique Crashes</th>
<th>Synthesized PoVs</th>
</tr>
</thead>
<tbody>
<tr>
<td>TensorFlow</td>
<td>412</td>
<td>103</td>
<td>86 / 103 (83%)</td>
</tr>
<tr>
<td>PyTorch</td>
<td>747</td>
<td>81</td>
<td>49 / 81 (60%)</td>
</tr>
<tr>
<td>All</td>
<td>1159</td>
<td>184</td>
<td>135 / 184 (73%)</td>
</tr>
</tbody>
</table>

- IvySyn synthesized 135 PoVs and was attributed with 39 CVEs.
Q3: Which *IvySyn* mutations are the most successful in uncovering memory errors?

<table>
<thead>
<tr>
<th>IvySyn Type-aware Mutation Type</th>
<th>Total PoVs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tensors with random dimension sizes</td>
<td>46</td>
</tr>
<tr>
<td>Tensors with extreme values</td>
<td>25</td>
</tr>
<tr>
<td>Permutations of original arguments</td>
<td>19</td>
</tr>
<tr>
<td>Zero values</td>
<td>12</td>
</tr>
<tr>
<td>Lists with extreme values</td>
<td>13</td>
</tr>
<tr>
<td>Tensors with empty shape</td>
<td>8</td>
</tr>
<tr>
<td>Extreme values in primitive types</td>
<td>6</td>
</tr>
<tr>
<td>Empty lists</td>
<td>3</td>
</tr>
<tr>
<td>Deep tensors</td>
<td>3</td>
</tr>
</tbody>
</table>
Evaluation ➔ Effectiveness per Mutation Type

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

<table>
<thead>
<tr>
<th>IvySyn Type-aware Mutation Type</th>
<th>Total PoVs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tensors with random dimension sizes</td>
<td>46</td>
</tr>
<tr>
<td>Tensors with extreme values</td>
<td>25</td>
</tr>
<tr>
<td>Permutations of original arguments</td>
<td>19</td>
</tr>
<tr>
<td>Zero values</td>
<td>12</td>
</tr>
<tr>
<td>Lists with extreme values</td>
<td>13</td>
</tr>
<tr>
<td>Tensors with empty shape</td>
<td>8</td>
</tr>
<tr>
<td>Extreme values in primitive types</td>
<td>6</td>
</tr>
<tr>
<td>Empty lists</td>
<td>3</td>
</tr>
<tr>
<td>Deep tensors</td>
<td>3</td>
</tr>
</tbody>
</table>

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

Evaluation

Conclusion
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/

neophytos_christou@brown.edu (Brown University)  IvySyn
Conclusion

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

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

neophytos_christou@brown.edu (Brown University)  IvySyn
Conclusion

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