Responding to the Trillion-Dollar Threat of Cybercrime

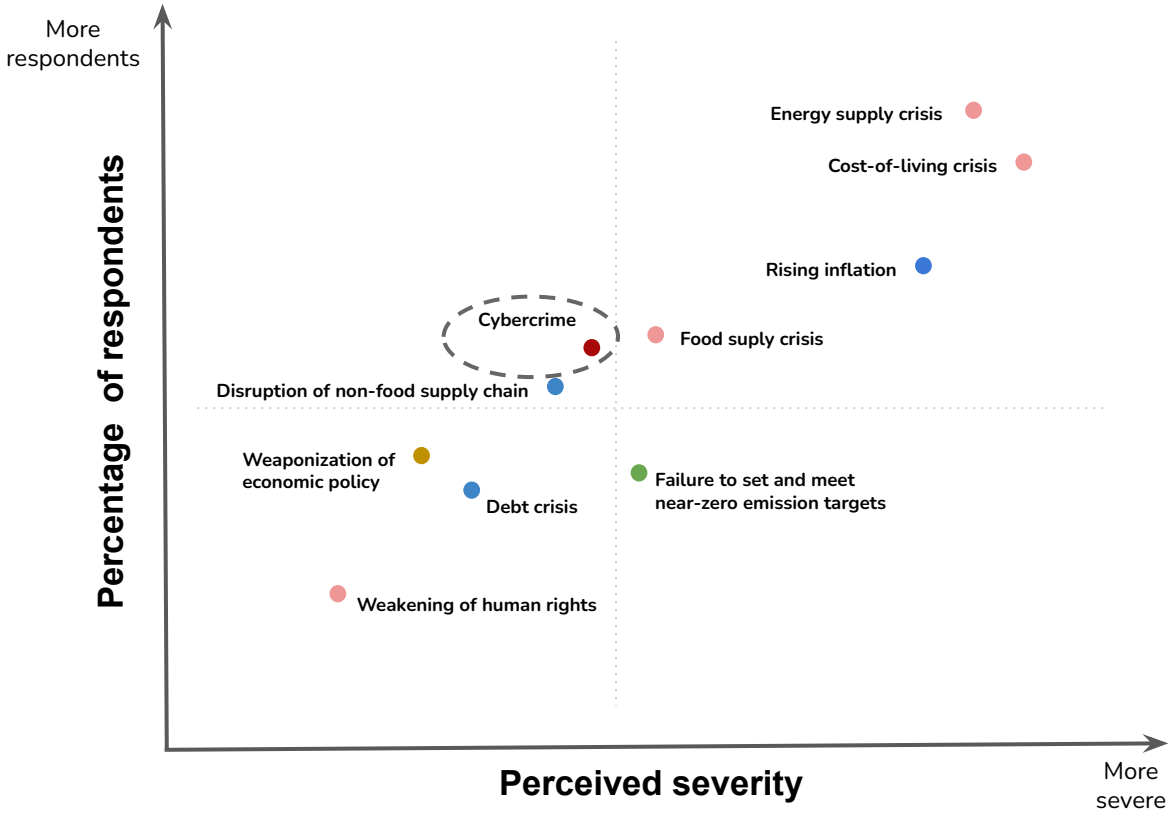
Vaggelis Atlidakis

BROWN

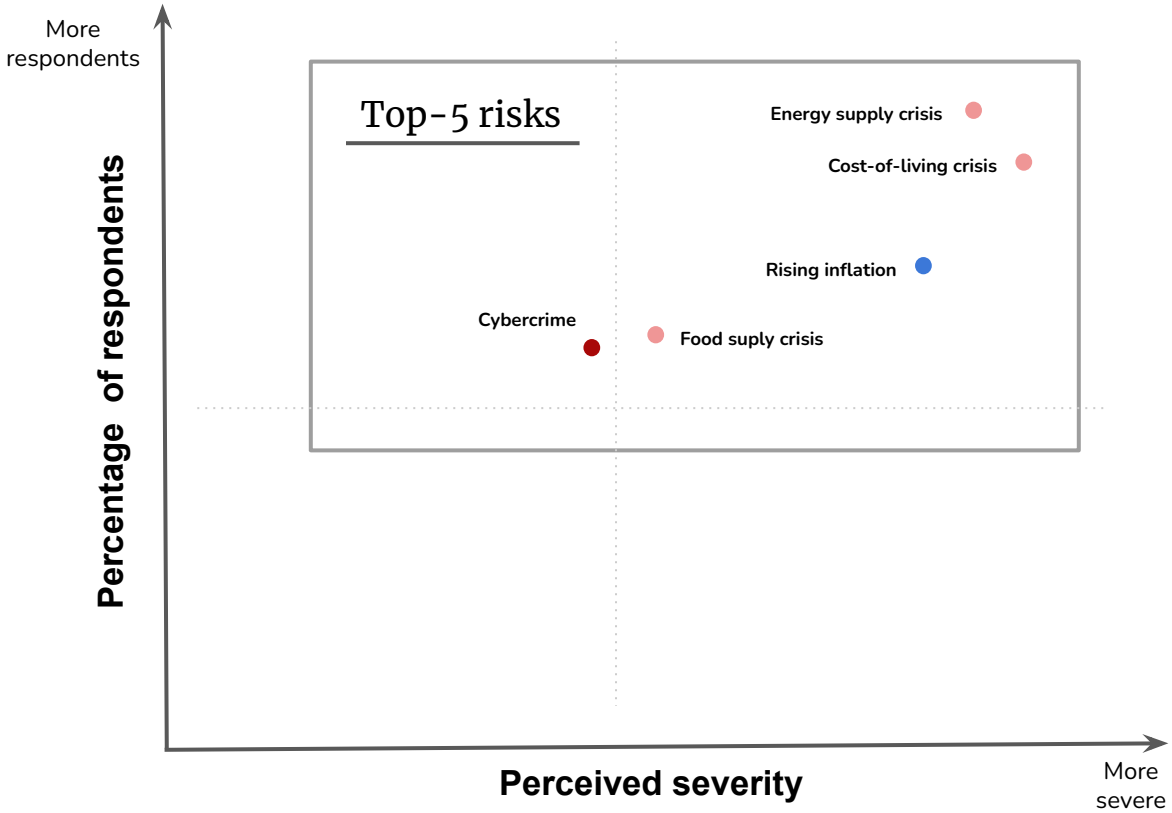
LACEWORK


WORLD
ECONOMIC
FORUM

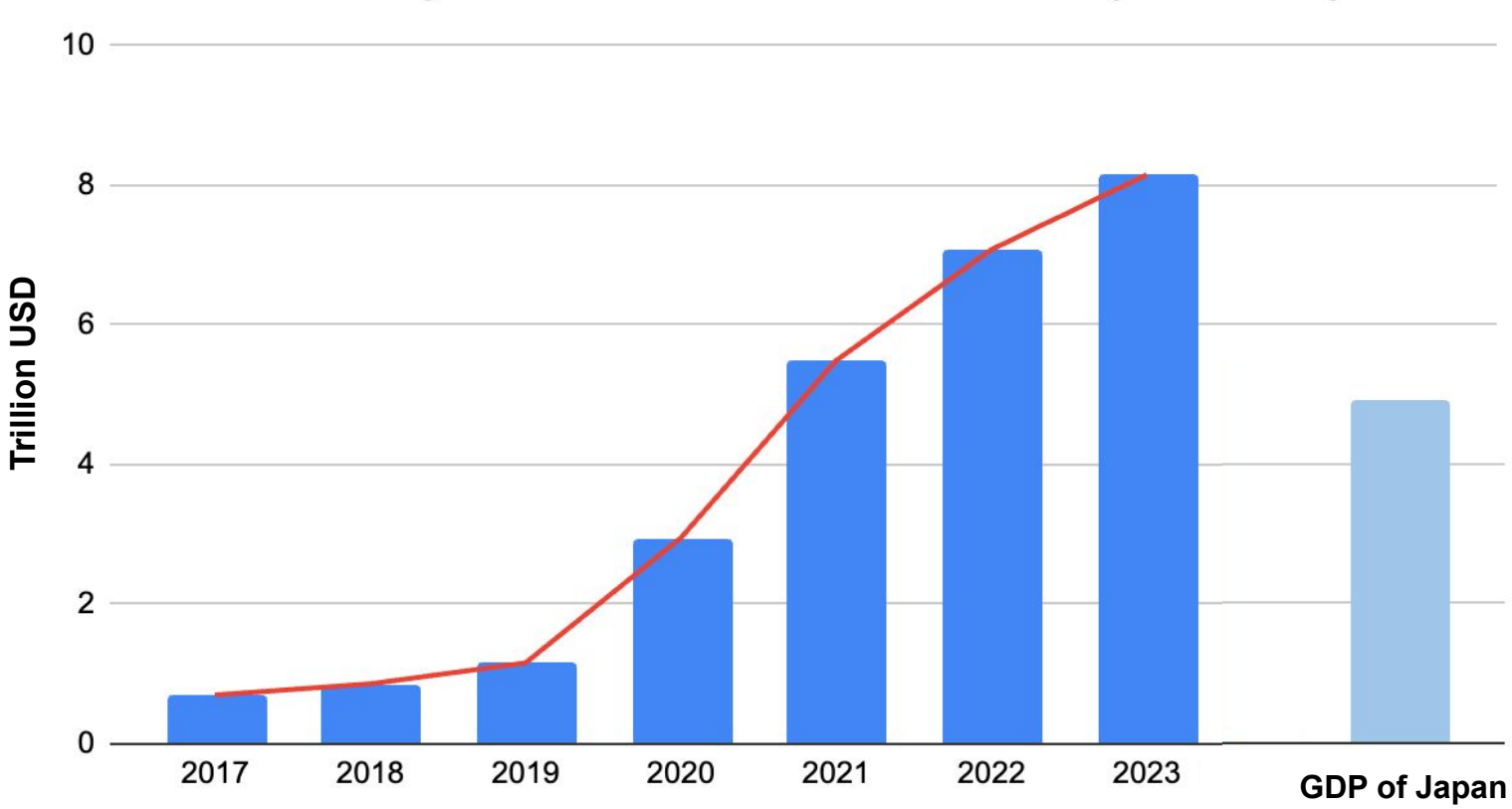
Global Manifesting Risks, World Economic Forum 2022-23



Global Manifesting Risks, World Economic Forum 2022-23



Estimated Cost of Cybercrime, Worldwide 2017-23 (Statistica)



What should we do?

Goal

- Attack surface reduction

Our Arsenal

- Software Testing: *Investigate whether a piece of software manifests undesirable behavior (bugs)*
- Software Hardening: *Enforce the principle of least privilege*
- Risk Management: *Decide what vulnerabilities to focus on (patch) first*

Goal

- Attack surface reduction

Outline

- Software Testing
  - Fairtest [Euro S&P '17]
  - RESTler [ICSE ’19, ICST '20]
  - Cynthia [ICSE ’21]
  - IvySyn [USENIX SEC '23]
- Software Hardening
  - PixelDP [S&P '19]
  - EPF [USENIX ATC '23]
  - FineIBT [RAID '23]
  - Sysxchg [CCS '23]
- Risk Management
  - Lacework

Goal

- Attack surface reduction

Outline

- Software Testing
  - Fairtest [Euro S&P '17]
  - **RESTler [ICSE '19, ICST '20]**
  - Cynthia [ICSE '21]
  - **IvySyn [USENIX SEC '23]**
- Software Hardening
  - PixelDP [S&P '19]
  - EPF [USENIX ATC '23]
  - FineIBT [RAID '23]
  - **Sysxchg [CCS '23]**
- Risk Management
  - **Lacework**

Goal

- ➢ Attack surface reduction

Outline

- ➢ Software Testing
  - Fairtest [Euro S&P '17]
  - ⇒ **RESTler [ICSE '19, ICST '20]**
  - Cynthia [ICSE '21]
  - **IvySyn [USENIX SEC '23]**
- ➢ Software Hardening
  - PixelDP [S&P '19]
  - EPF [USENIX ATC '23]
  - FineIBT [RAID '23]
  - **Sysxchg [CCS '23]**
- ➢ Risk Management
  - **Lacework**

What

- 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

Example

A Gitlab test

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

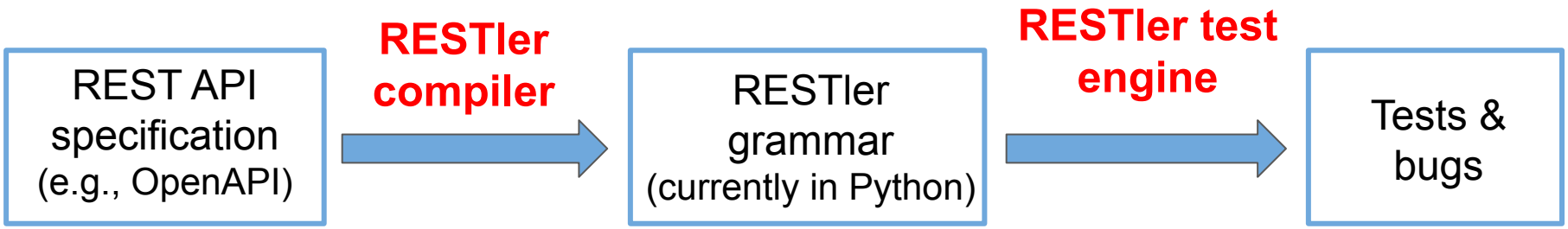
```
curl --request POST --header 'PRIVATE-TOKEN: <your_access_token>' \
     --header "Content-Type: application/json" \
     --data '{"branch": "main", "author_email": "author@example.com", "author_name": "Firstname Lastname",
              "content": "some content", "commit_message": "create a new 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"
```

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: Sample bugs?

Study subject: Gitlab

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

➢ …and more in production Azure cloud services

Checking Security Properties of RESTful Cloud Services

What

- Find errors that don't cause ``500 Internal Server Errors''

Why

- Not everything is visible from response status codes
- Security-relevant issues may be going unnoticed

How: *Augment Stateful REST API Fuzzing*

- Introduce security rules and check for violations

Selected security rules

- Use-after-free rule
  1. Delete /api/projects/1
  - ➢ Access /api/projects/1 **MUST FAIL**

- Resource-hierarchy rule
  1. Create /api/projects/1 and /api/projects/2
  2. Create /api/projects/1/branches/1
  - ➢ Access /api/projects/2/branches/1 **MUST FAIL**

- Resource-leakage rule
  1. Create /api/projects/1 and receive error code (e.g., 404 or 500 HTTP status)
  - ➢ Access /api/projects/1 **MUST FAIL**

Selected security rules

- Use-after-free rule
  1. Delete /api/projects/1
  - ➢ Access /api/projects/1 **MUST FAIL**
- Resource-hierarchy rule
  1. Create /api/projects/1 and /api/projects/2
  2. Create /api/projects/1/branches/1
  - ➢ Access /api/projects/2/branches/1 **MUST FAIL**
- Resource-leakage rule
  1. Create /api/projects/1 and receive error code (e.g., 404 or 500 HTTP status)
  - ➢ Access /api/projects/1 **MUST FAIL**

We've just augmented stateful REST API fuzzing with active property checking!

Outline

- ➢ Software Testing
  - ○ Fairtest [Euro S&P '17]
  - ✓ **RESTler [ICSE '19, ICST '20]**
  - ○ Cynthia [ICSE '21]
  - ⇒ **IvySyn [USENIX SEC '23]**
- ➢ Software Hardening
  - ○ PixelDP [S&P '19]
  - ○ EPF [USENIX ATC '23]
  - ○ FineIBT [RAID '23]
  - ○ **Sysxchg [CCS '23]**
- ➢ Vulnerability Risk Management
  - ○ **Lacework**

What

- 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);
  // ...
```

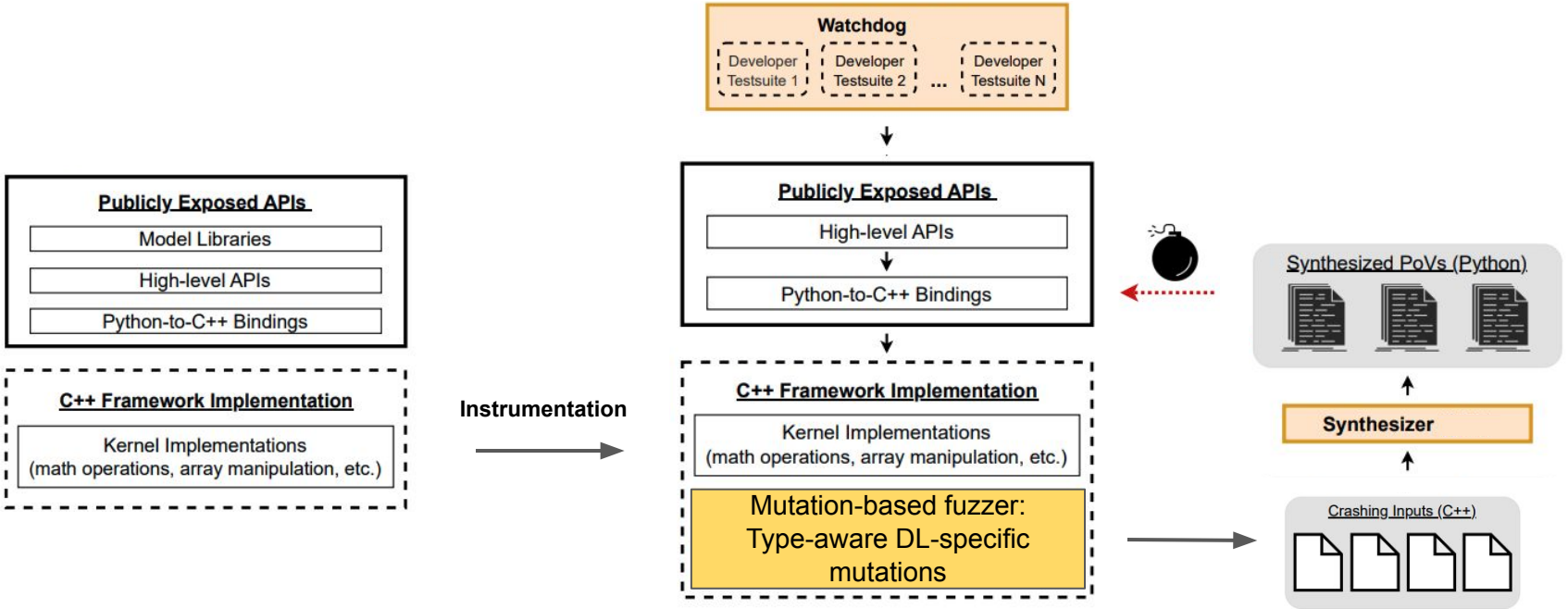
Triggers a crash

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)
```

Overview


Watchdog
Developer Testsuite 1
Developer Testsuite 2
...
Developer Testsuite N
Publicly Exposed APIs
Model Libraries
High-level APIs
Python-to-C++ Bindings
C++ Framework Implementation
Kernel Implementations (math operations, array manipulation, etc.)
Instrumentation
Publicly Exposed APIs
High-level APIs
Python-to-C++ Bindings
C++ Framework Implementation
Kernel Implementations (math operations, array manipulation, etc.)
Mutation-based fuzzer: Type-aware DL-specific mutations
Synthesized PoVs (Python)
Synthesizer
Crashing Inputs (C++)

Selected evaluation results

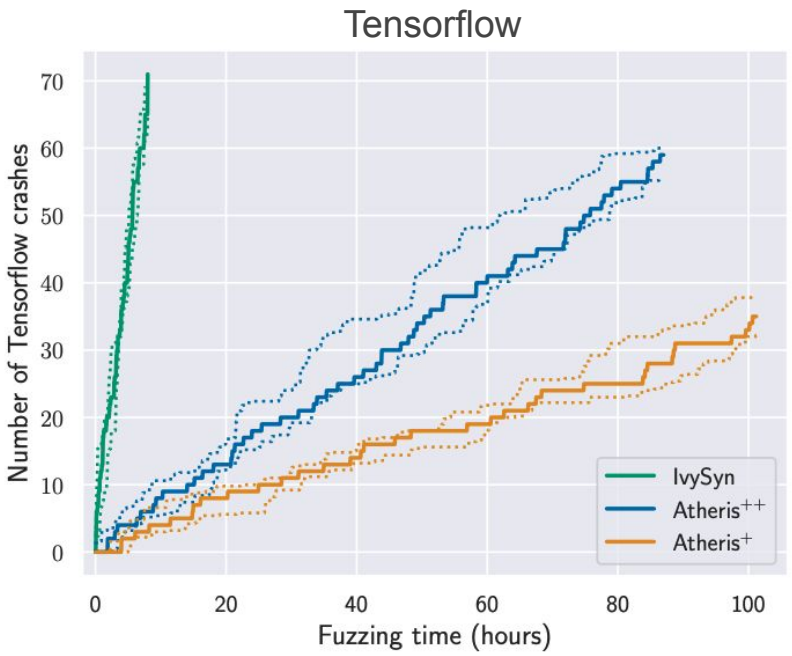
- Q1: How quickly can IvySyn find crashes?
- Q2: How effective is IvySyn in synthesizing PoVs?

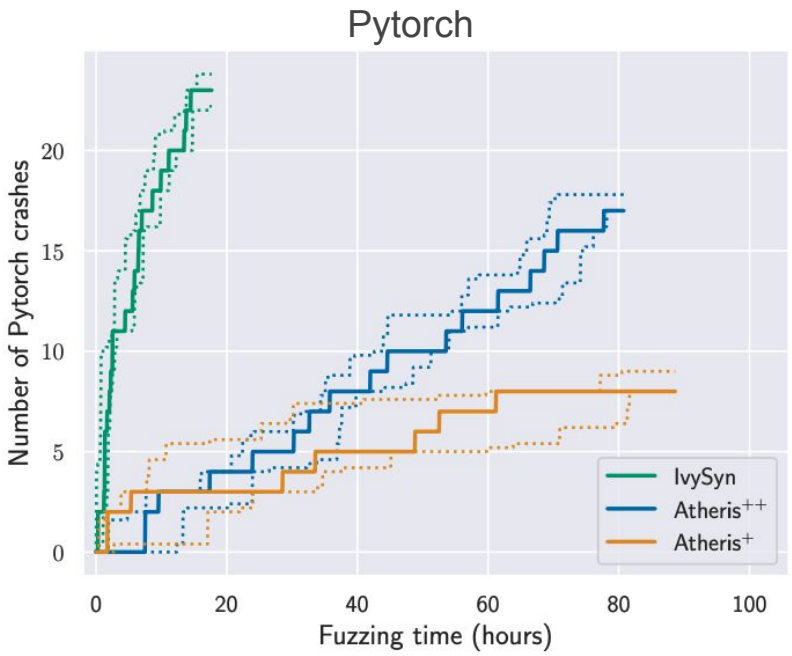
Target Frameworks: Tensorflow and Pytorch

Against *Atheris*: https://github.com/google/atheris

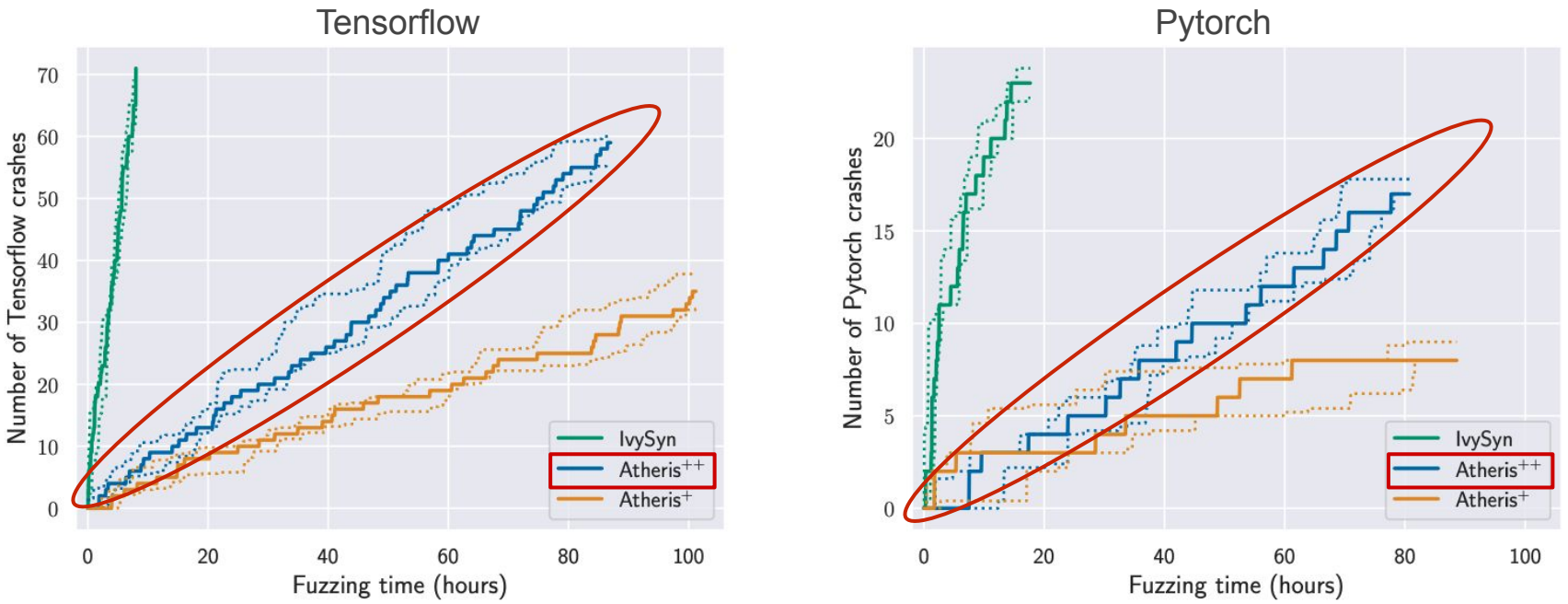
- ***Atheris*+**: Default *Atheris* + automation
- **Atheris++**: Default *Atheris* + automation + type-aware
- **IvySyn:** Type-aware + DL-specific mutations

Efficiency in finding crashing inputs (Q1)

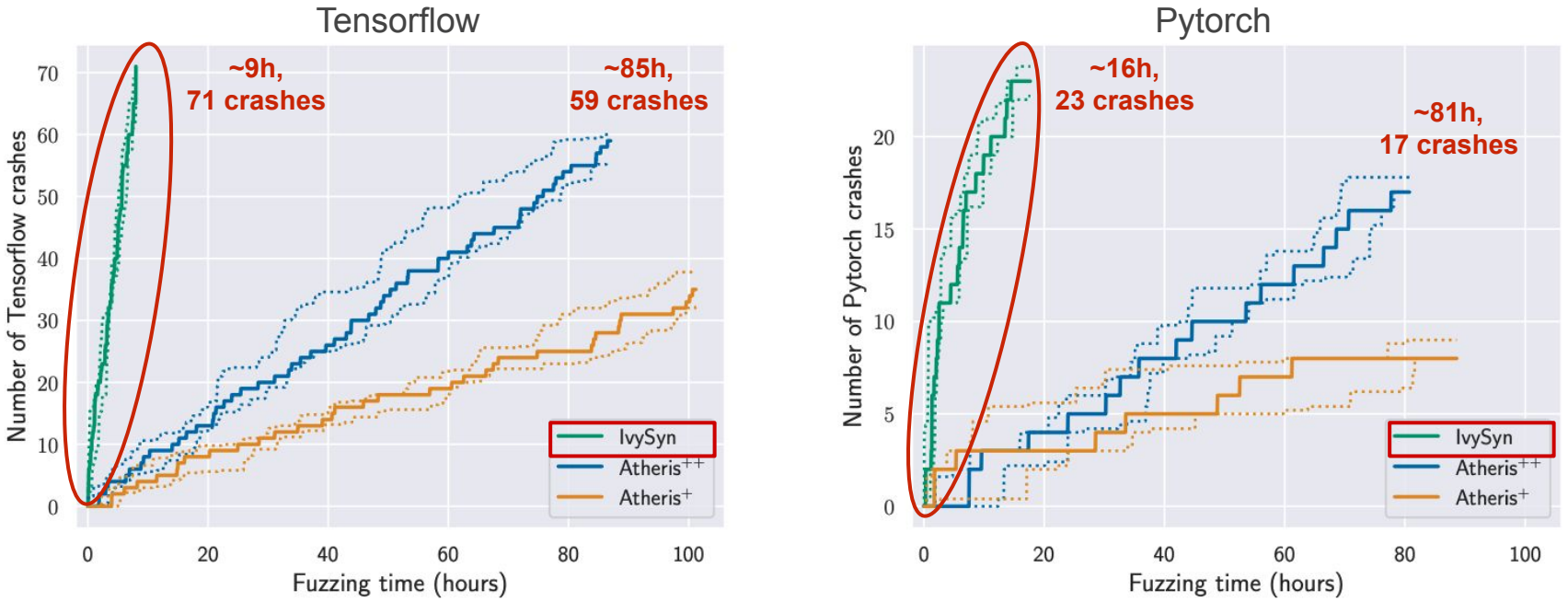




Efficiency in finding crashing inputs (Q1)



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 unique 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);
  // ...
```

Outline

- ➢ Software Testing
  - Fairtest [Euro S&P '17]
  - ✓ **RESTler [ICSE '19, ICST '20]**
  - Cynthia [ICSE '21]
  - ✓ **IvySyn [USENIX SEC '23]**
- ➢ Software Hardening
  - PixelDP [S&P '19]
  - EPF [USENIX ATC '23]
  - FineIBT [RAID '23]
  - **Sysxchg [CCS '23]**
- ➢ Vulnerability Risk Management
  - **Lacework**

Outline

- ➢ Software Testing
  - ○ Fairtest [Euro S&P '17]
  - ✓ **RESTler [ICSE '19, ICST '20]**
  - ○ Cynthia [ICSE '21]
  - ✓ **IvySyn [USENIX SEC '23]**
- ➢ Software Hardening
  - ○ PixelDP [S&P '19]
  - ○ EPF [USENIX ATC '23]
  - ○ FineIBT [RAID '23]
  - ○ **Sysxchg [CCS '23]**
- ➢ Vulnerability Risk Management
  - ○ **Lacework**

↓ Top down (start from the APIs)

↑ Bottom up (start from native implementations)

Outline

- ➢ Software Testing
  - ○ Fairtest [Euro S&P '17]
  - ✓ **RESTler [ICSE '19, ICST '20]**
  - ○ Cynthia [ICSE '21]
  - ✓ **IvySyn [USENIX SEC '23]**
- ➢ Software Hardening
  - ○ PixelDP [S&P '19]
  - ○ EPF [USENIX ATC '23]
  - ○ FineIBT [RAID '23]
  - ⇒ **Sysxchg [CCS '23]**
- ➢ Vulnerability Risk Management
  - ○ **Lacework**

What

- A process must not have access to more syscalls than it needs

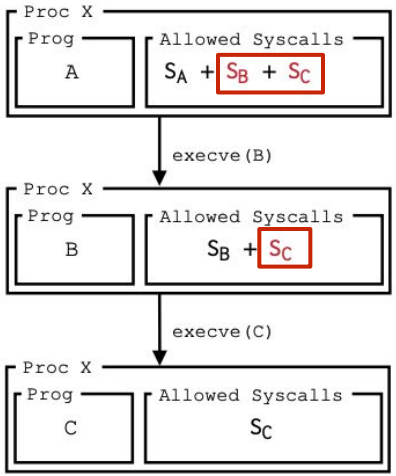
Why

- Linux kernel v6.0.8 provides ~400 sycalls
- Seccomp-BPF approaches: parent overpriviledge

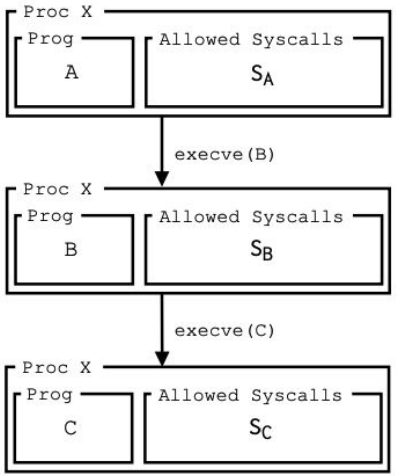
How: Process-specific syscall filters

Overview

Inherited filters

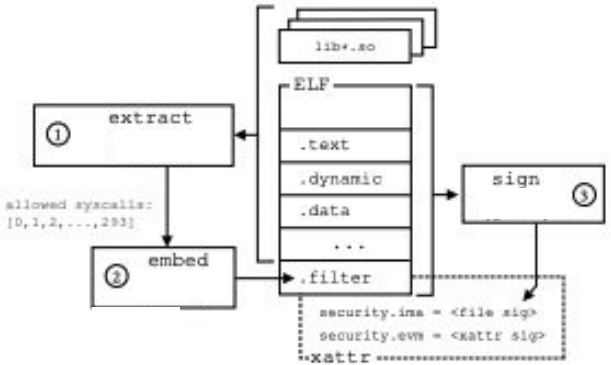


Adaptive *exec* filters

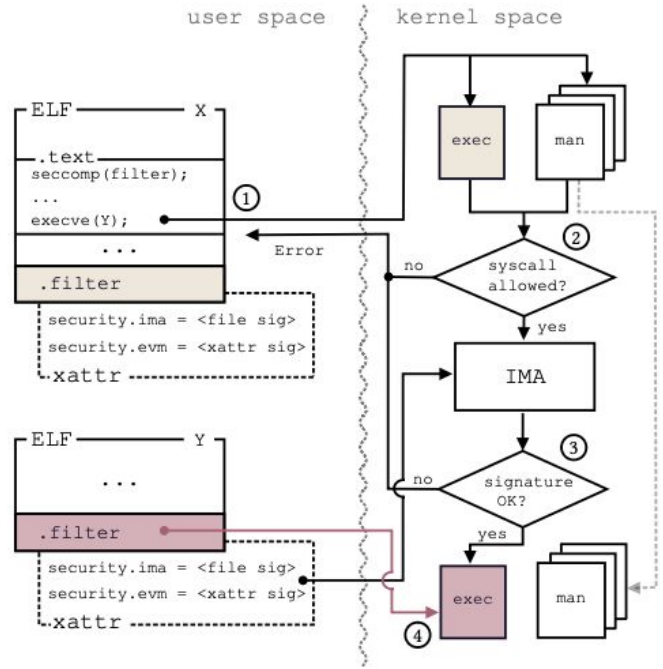


SysXCHG components

Offline phase



Online phase



https://gitlab.com/brown-ssl/sysxchg (Linux kernel v6.0.8, ≈ 600 LOC in C)

Selected results on the PaSh benchmark

Benchmarks	No of children	*Inhertance* model	*SysXCHG* model	Overpriviledge reduction	Unnecessary funct. (examples)	Critical
bi-grams	11	122	84	45%	{c,d,r}path chown	✓
shortest-scripts	8	108	84	28%	fattr id proc	✓
…	…	…	…	…	…	…
→	1	60	51	17%	{c,tmp}path proc	✓
AUR Package Compilation	69	176	84	109%	protexec vminfo …	✓
….	…	…	…	…	…	…
→	3	123	74	66%	{c,d,r,w}path chown …	✓
Wikipedia Web Indexing	16	146	84	73%	chown fattr protexec	✓

*PaSh: Light-touch Data-Parallel Shell Processing [Vasilakis et al., EuroSys '21]

Selected results on the PaSh benchmark

Benchmarks	No of children	*Inhertance* model	*SysXCHG* model	Overpriviledge reduction	Unnecessary funct. (examples)	Critical
bi-grams	11	122	84	45%	{c,d,r}path chown	✓
shortest-scripts	8	108	84	28%	fattr id proc	✓
…	…	…	…	…	…	…
→	1	60	51	17%	{c,tmp}path proc	✓
AUR Package Compilation	69	176	84	109%	protexec vminfo …	✓
….	…	…	…	…	…	…
→	3	123	74	66%	{c,d,r,w}path chown …	✓
Wikipedia Web Indexing	16	146	84	73%	chown fattr protexec	✓

➢ CPU overhead is ~1.7%

*PaSh: Light-touch Data-Parallel Shell Processing [Vasilakis et al., EuroSys '21]

Outline

- ➢ Software Testing
  - ○ Fairtest [Euro S&P '17]
  - ✓ **RESTler [ICSE '19, ICST '20]**
  - ○ Cynthia [ICSE '21]
  - ✓ **IvySyn [USENIX SEC '23]**
- ➢ Software Hardening
  - ○ PixelDP [S&P '19]
  - ○ EPF [USENIX ATC '23]
  - ○ FineIBT [RAID '23]
  - ✓ **Sysxchg [CCS '23]**
- ➢ Vulnerability Risk Management

  ⇒ ~~**Lacework**~~

Disclaimer: This work is a Lacework product, led by Patrice Godefroid and Geremy Condra, and was started before I joined Lacework.

https://www.lacework.com/blog/why-active-vulnerability-detection-is-a-game-changer-for-vulnerability-risk-management/

What

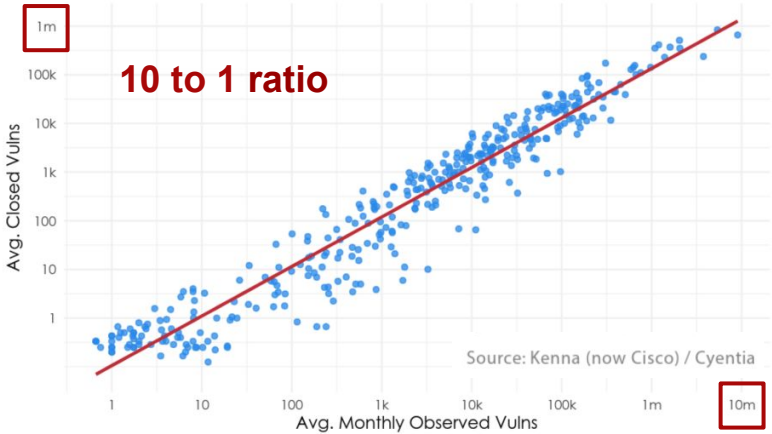
- Select what vulnerabilities to patch first

Why

- We cannot patch everything

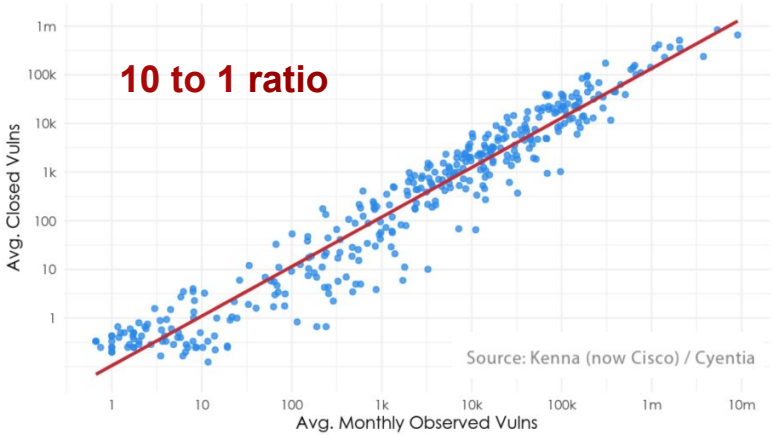
The drama of unpatched vulnerabilities

Accumulation velocity

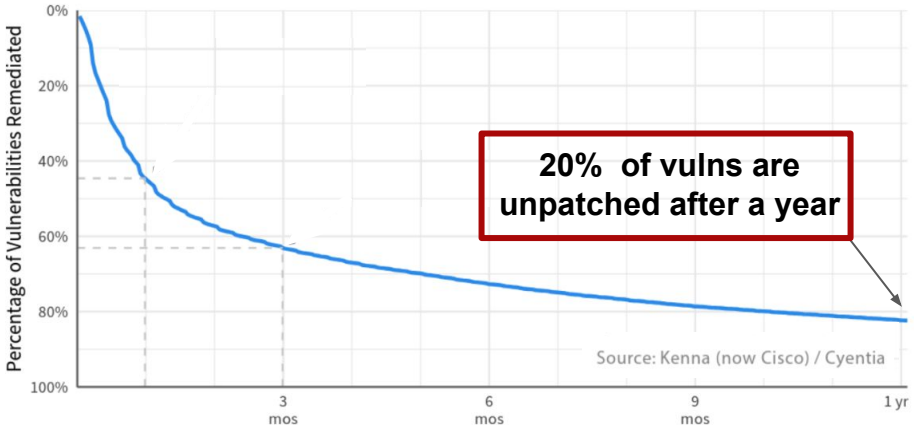


The drama of unpatched vulnerabilities

Accumulation velocity



Remediation velocity



What

- Select what vulnerabilities to patch first

Why

- We cannot patch everything

How: *Active Vulnerability Management*

Results on package (in) activity

➢ **Study Subject:** ~2.5K host machines over 2 weeks

❖ Unique host packages: 1361
  I. Vulnerable: **478**
  II. Active: 235
  III. Vulnerable & Active: **98**

❖ Active vulnerability management: From 478 to 98 there is a **79%** reduction
  ➢ Monitor package activity in production

https://www.lacework.com/blog/why-active-vulnerability-detection-is-a-game-changer-for-vulnerability-risk-management/

- ➢ Software Testing
  - Fairtest [Euro S&P '17]
  - ✓ **RESTler [ICSE '19, ICST '20]**
  - Cynthia [ICSE '21]
  - ✓ **IvySyn [USENIX SEC '23]**
- ➢ Software Hardening
  - PixelDP [S&P '19]
  - EPF [USENIX ATC '23]
  - FineIBT [RAID '23]
  - ✓ **Sysxchg [CCS '23]**
- ➢ Vulnerability Risk Management
  - ✓ **Lacework**

- ➢ Software Testing
  - ○ Fairtest [Euro S&P '17]
  - ✓ **RESTler [ICSE '19, ICST '20]**
  - ✓ Cynthia [ICSE '21]
  - ✓ **IvySyn [USENIX SEC '23]**
- ➢ Software Hardening
  - ○ PixelDP [S&P '19]
  - ✓ EPF [USENIX ATC '23]
  - ✓ FineIBT [RAID '23]
  - ✓ **Sysxchg [CCS '23]**
- ➢ Vulnerability Risk Management
  - ✓ **Lacework**

- ➢ Software Testing
  - ✓ Fairtest [Euro S&P '17]
  - ✓ **RESTler [ICSE '19, ICST '20]**
  - ✓ Cynthia [ICSE '21]
  - ✓ **IvySyn [USENIX SEC '23]**
- ➢ Software Hardening
  - ✓ PixelDP [S&P '19]
  - ✓ EPF [USENIX ATC '23]
  - ✓ FineIBT [RAID '23]
  - ✓ **Sysxchg [CCS '23]**
- ➢ Vulnerability Risk Management
  - ✓ **Lacework**

- Software Testing
  - ✓ Fairtest [Euro S&P '17]
  - ✓ **RESTler [ICSE '19, ICST '20]**
  - ✓ Cynthia [ICSE '21]
  - ✓ **IvySyn [USENIX SEC '23]**
- Software Hardening
  - ✓ PixelDP [S&P '19]
  - ✓ EPF [USENIX ATC '23]
  - ✓ FineIBT [RAID '23]
  - ✓ **Sysxchg [CCS '23]**
- Vulnerability Risk Management
  - ✓ **Lacework**

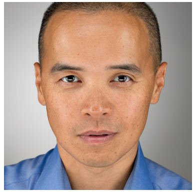
Goal

- Attack surface reduction

Thanks to all of them !!!




















金迪