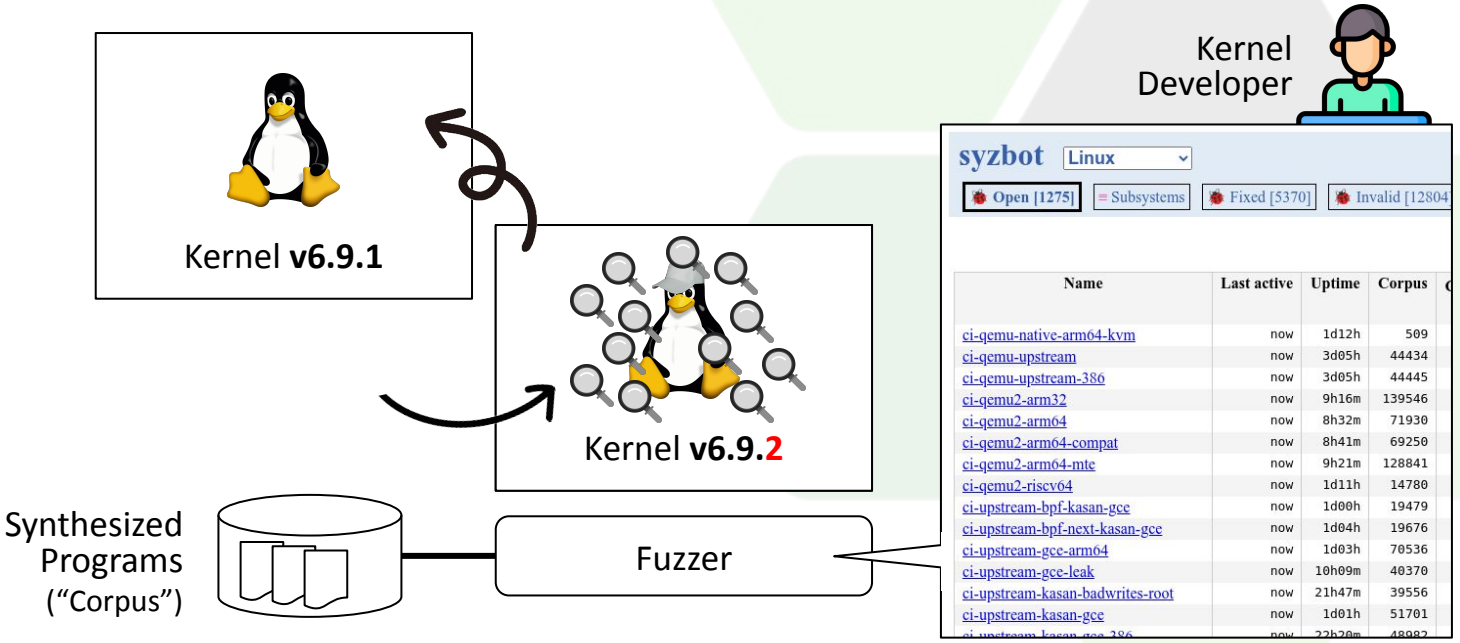
SyzRisk: A Change-Pattern-Based Continuous Kernel Regression Fuzzer

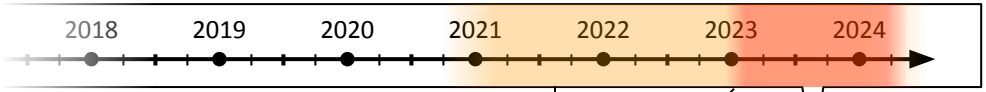
Gwangmu Lee[1], Duo Xu[1], Solmaz Salimi[2], Byoungyoung Lee[3], Mathias Payer[1]

[1] EPFL/HexHive
[2] Sharif University of Technology
[3] Seoul National University

Kernels are Tested **Continuously**

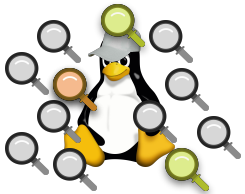


But, Most Bugs are from **Recent** Changes



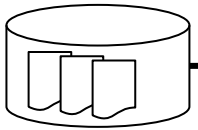
>50% CVEs within 3~4 years
[Alexopoulos et al., USENIX'22]

~75% crashes less than 1 year
(crashes from Syzkaller)



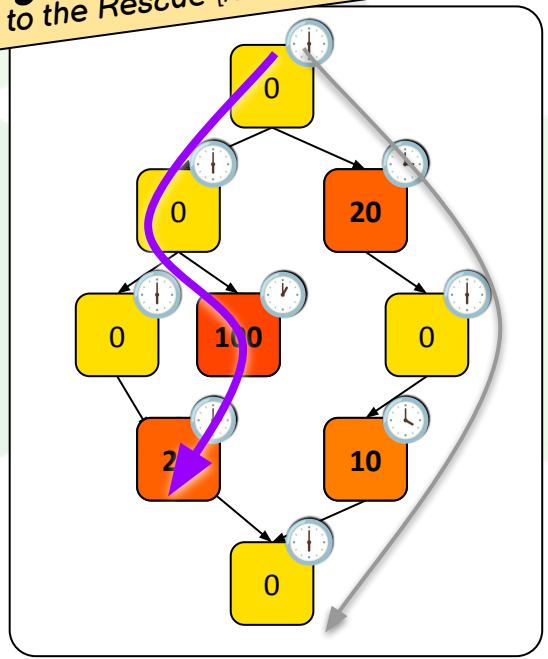
Kernel **v6.9.2**

Synthesized Programs ("Corpus")



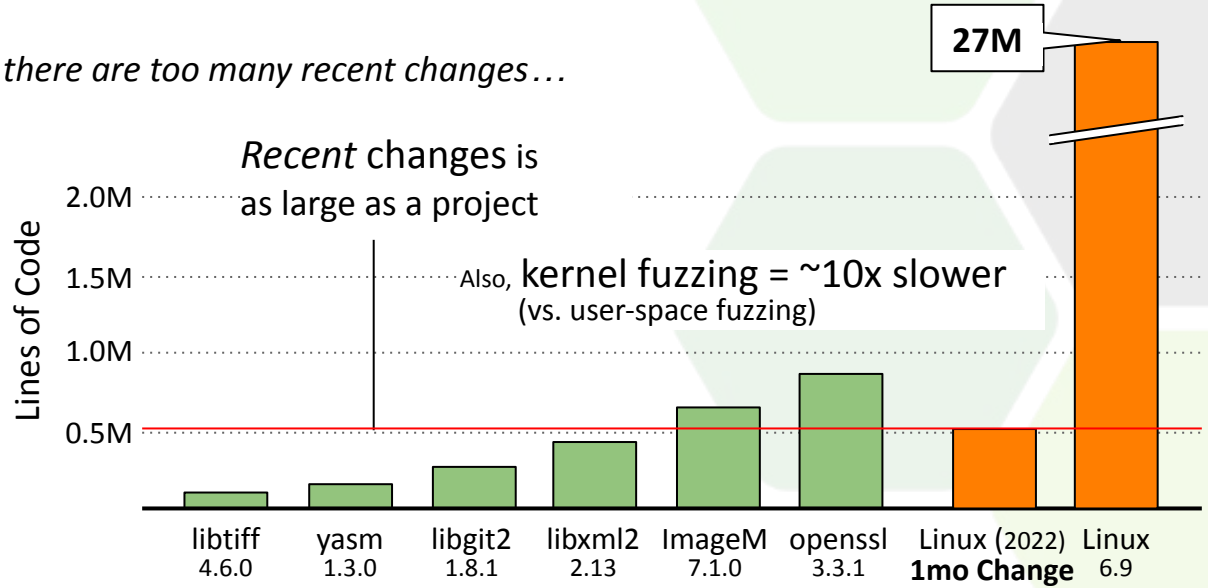
Fuzzer

Regression Fuzzing to the Rescue [AFLChurn]



Bad News: Regression Fuzzing **Does Not Scale to Kernel**

Because there are too many recent changes…



"Recent changes will be replaced by <u>new</u> recent changes before tested"

Wait. Are All Recent Changes **Equally** Risky?

Exhibit **A.**

```
- if (cpu->event == NULL) {
+ if (!cpu->event) {
    pr_err("...\n");
```

```
- printk("a=%u\n", a);
+ printk("a=%u, b=%u\n",
+         a, b);
```

✔ Doesn't involve much complexity.
✔ Devs likely consider all side-effects.

Exhibit **B.**

```
  struct memslots {
-     slot_t slots[MAX];
+     slot_t *slots;
  };
  ...
💀 *memslots->slots[0];
```

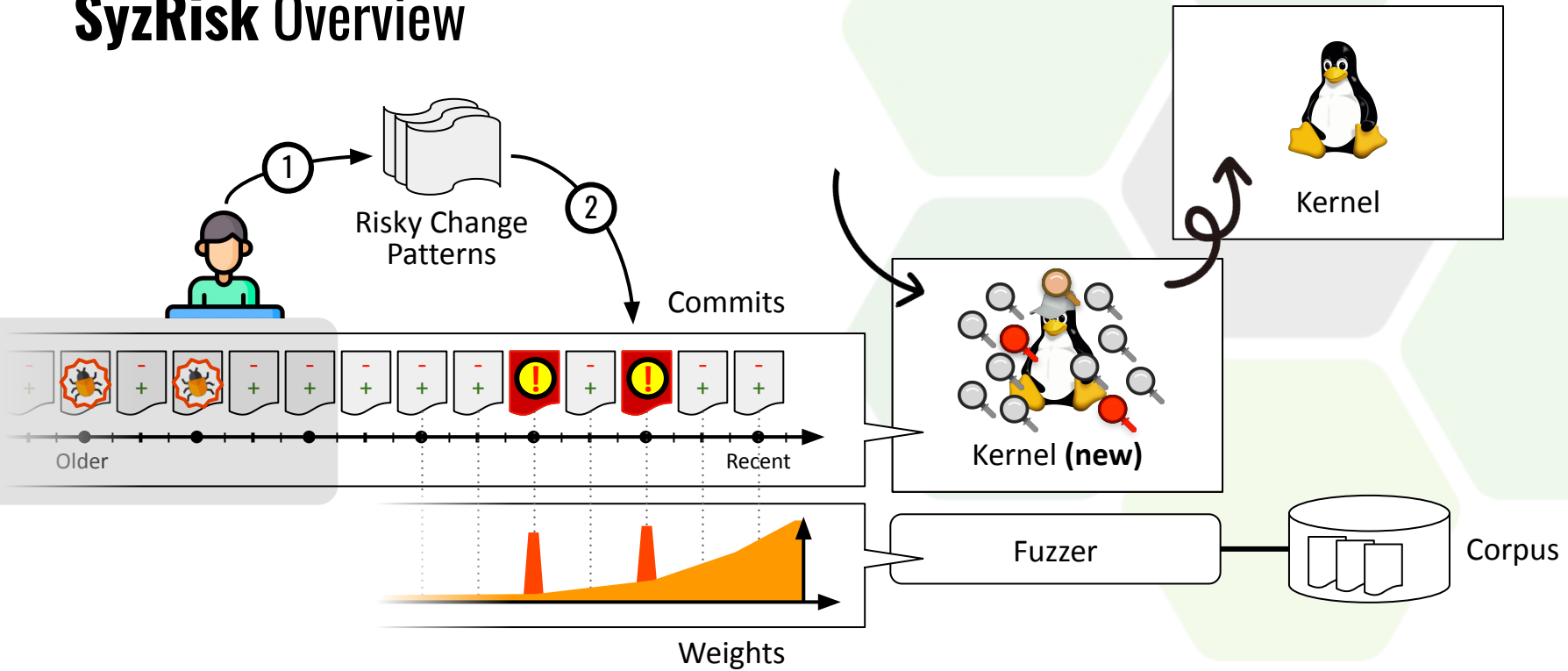
```
  conn = kmalloc();
💀 conn->path = kmalloc();

  if (is_outgoing) {
    ...
+   kfree(conn);
+   goto err;
```

🔥 May cause much complexity, possibly a global one.
💀 Developers will likely miss some side-effects.

Let's salvage regression fuzzing by weighting risky changes.

SyzRisk Overview



Collecting Risky Change Patterns

"Devs will notice some **recurring patterns** while fixing bugs."
"**Emulate what devs would do.**"

- Step 1: **investigate** known root causes and their fixes.
  Total **146** Linux root cause/fix pairs between **2020~2021**.
  Collected suspicious recurring change patterns. ("draft patterns," so to say)
- Step 2: collect **ground-truth** root causes and benign changes.
  Leveraged the Linux kernel commit convention (i.e., `FIXED: <commit_id>` in comment)
- Step 3: **calculate** the risk of patterns & **refine** them.
  E.g., splitting, elaborating, or dropping some draft patterns.

Example Collected Risky Patterns

```
        mutex_unlock(&hfi1_mutex);
        hfi1_free_ctxtdata(dd, uctxt);
done:
+       mmdrop(fdata->mm);
        kobject_put(&dd->kobj);
        kfree(fdata);
        return 0;
```

CVE-2020-27835 (Use-after-free)

Pattern: **Inside GOTO**

🔥 GOTOs are used for exception handling.
⇒ 💀 Mistakes easily lead to resource bug.

Full 23 patterns in paper & artifact repository

```
 static int sctp_setsockopt_delayed_ack(struct sock *sk,
-                                       char __user *optval, unsigned int o
+                                       struct sctp_sack_info *params,
+                                       unsigned int optlen)
 {
        struct sctp_sock *sp = sctp_sk(sk);
        struct sctp_association *asoc;
-       struct sctp_sack_info params;

        if (optlen == sizeof(struct sctp_sack_info)) {
-               if (copy_from_user(&params, optval, optlen))
-                       return -EFAULT;
-
-               if (params.sack_delay == 0 && params.sack_freq == 0)
+               if (params->sack_delay == 0 && params->sack_freq == 0)
                        return 0;
```

Crash on Jul 19th, 2020 (Buffer-overflow)

Pattern: **Pointer Promotion**

🔥 Variables change correct usage.
⇒ 💀 Devs may miss some code adjustment.

Q: Are Patterns **Generalizable**?

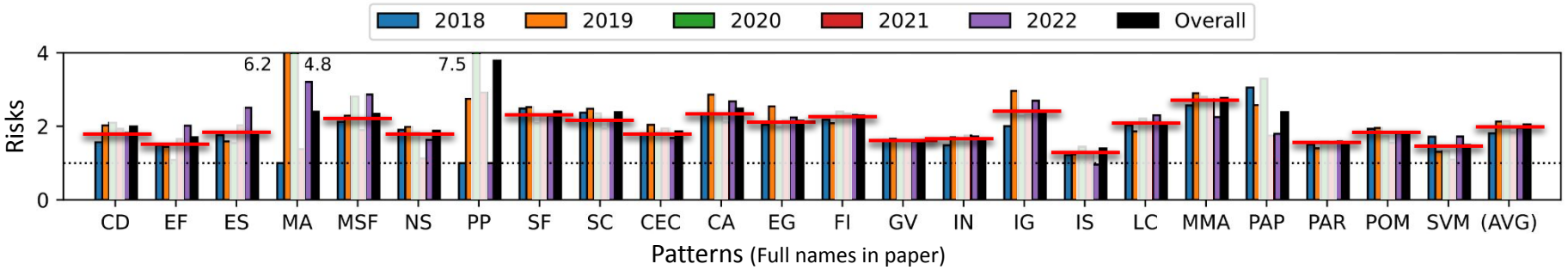


Figure. Pattern risks per year

- Ground-truth changes from each year.
- Pattern matching: Joern v1.360 and Python scripts.

✔ **Risks remain similar throughout time.**
✔ **Patterns remain risky regardless of when they were collected.**

Q: Do Patterns **Improve** Regression Fuzzing?

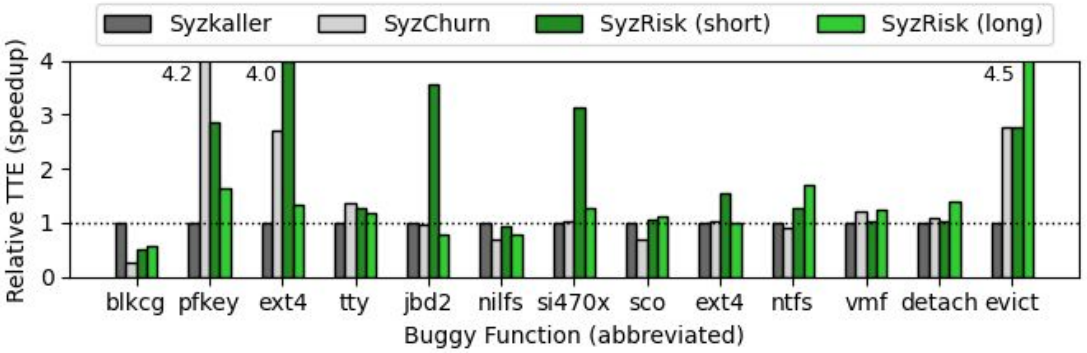


Figure. TTE Speedup Comparison

✔ **Quicker bug discovery: avg. 61% faster than Syzkaller (`short`)**

✔ **Longer "recentness" time (3~9x longer than SyzChurn)**

SyzRisk Implementation

- Based on Syzkaller
- Change-pattern-based weights
- SyzChurn = AFLChurn kernel port (recentness-based weights)

Evaluation Setting

- Kernel: Linux v6.0
- Three iteration average
- One iteration = 72 hours
- `short`/`long` = length of "recent" (`short` = 1mo, `long` = 3mo)

Conclusion

Kernel bugs are mostly caused by recent changes.

Regression fuzzing prioritizes them, but kernels have too many recent changes.

Intuition: "not every recent change is risky."

What we did:

- Collected risky patterns and showed their generality.
- Implemented **SyzRisk**, 61% TTE speedup.

Available in paper:

- Definition of *riskiness* of a change.
- Completeness of discovered bugs.
- …

Thank you

Presenter Gwangmu Lee

Artifact https://github.com/HexHive/SyzRisk

Backups

Defining Riskiness of Code Changes

Intuition: "**How likely** does a change pattern c cause problems?"

Formally,

$$\text{Risk } \boldsymbol{R(c)} := \frac{P(c \text{ in Root Causes})}{P(c \text{ in All Changes})} \approx \frac{\textit{freq}(c \text{ in } \textbf{Known} \text{ Root Causes})}{\textit{freq}(c \text{ in All Changes})}$$

Q: "What if there are **multiple changes** lumped together?"

$$\text{Risk } \boldsymbol{R(m)} := \frac{P(\forall c \text{ in Root Causes})}{P(\forall c \text{ in All Changes})} \approx \frac{\prod P(c \text{ in Root Causes})}{\prod P(c \text{ in All Changes})} = \prod \boldsymbol{R(c)}$$

Modification
(clustered changes)

Q: How Much Do Patterns **Highlight** Root Causes?

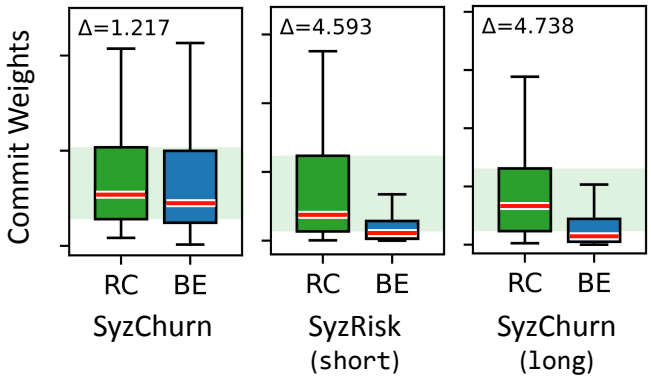
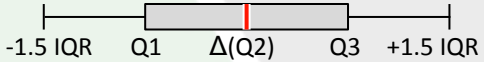


Figure. Commit weight distribution.
(RC: root-cause, BE: benign)

- Period: 2021.07 ~ 2022.06.
- Δ: **median** value.



✔ **3.7x to 3.9x** more highlight.
(based on median)

✔ **~95%** of root-cause matches.
(i.e., ~5% of unmatched false-negatives)

More periods in the paper

Q: How **Completely** Does SyzRisk Find Bugs?

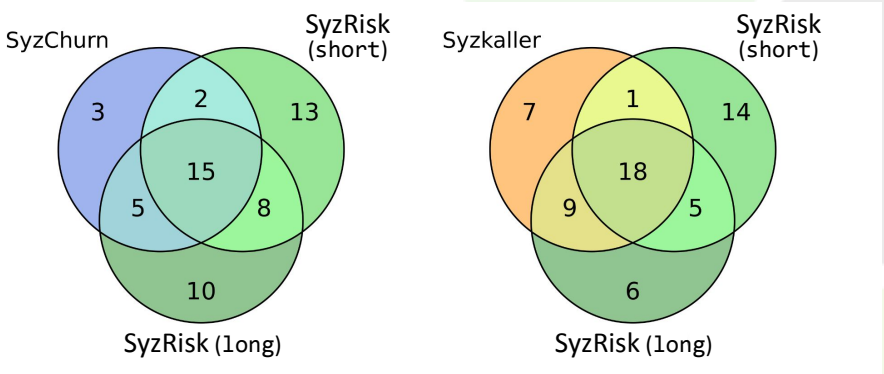


Figure. Number of bugs found by fuzzers.

- Bugs found at least once in all three trials.

✔ **85.5% of the bugs found by SyzRisk.** (`short+long`)
(5 of 7 Syzkaller-exclusive bugs were **duplicates** of found ones)