CRISP Grand Challenges Workshop 2023

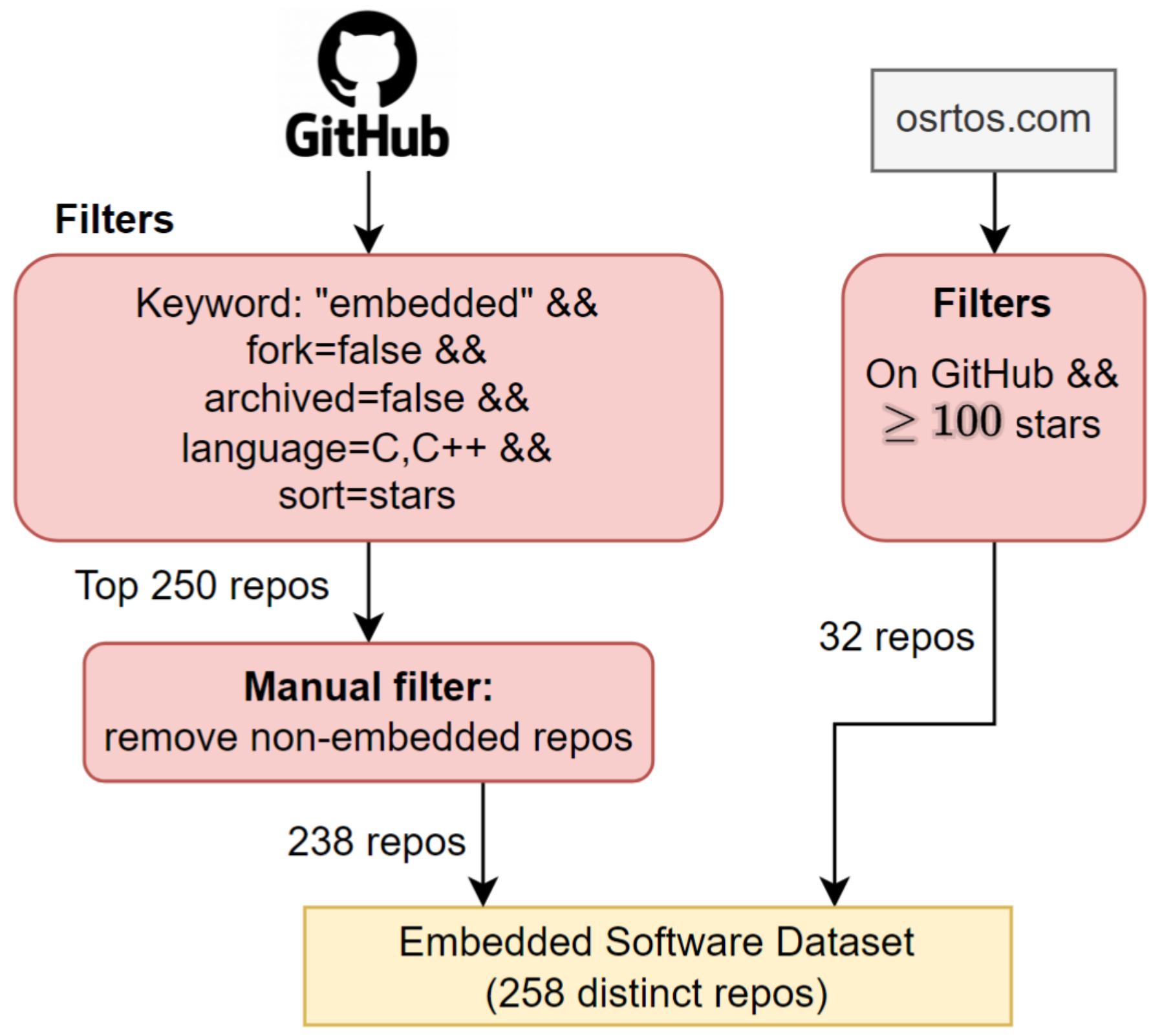
# An Empirical Study on the Use of Static Analysis Tools in Open Source Embedded Software

Mingjie Shen, Akul Pillai, Brian A. Yuan, James C. Davis, Aravind Machiry

## OVERVIEW

We investigate the use of Static Application Security Testing (SAST) tools in Open-Source Embedded Software (EMBOSS) projects used in safety-critical systems. We found the lack of SAST tool usage, with only 3% of projects employing them, citing ineffectiveness and false positives as reasons. We applied SAST tools and found GitHub's CodeQL to be the most effective, uncovering 540 defects, with 74% likely being security vulnerabilities. We recommend EMBOSS engineers adopt modern SAST tools for enhanced security.

## EMBOSS DATASET COLLECTION


GitHub
osrtos.com
Filters
Keyword: "embedded" && fork=false && archived=false && language=C,C++ && sort=stars
Filters
On GitHub && ≥ 100 stars
Top 250 repos
32 repos
Manual filter: remove non-embedded repos
238 repos
Embedded Software Dataset (258 distinct repos)

## RQ1: PREVALENCE OF SAST TOOLS

- Most (97%) of the EMBOSS repositories do not use SAST tools.
- Many EMBOSS repositories rely on compiler warnings instead of dedicated SAST tools.
- Most developers are aware of CI Workflows and use them to run their SAST tools.

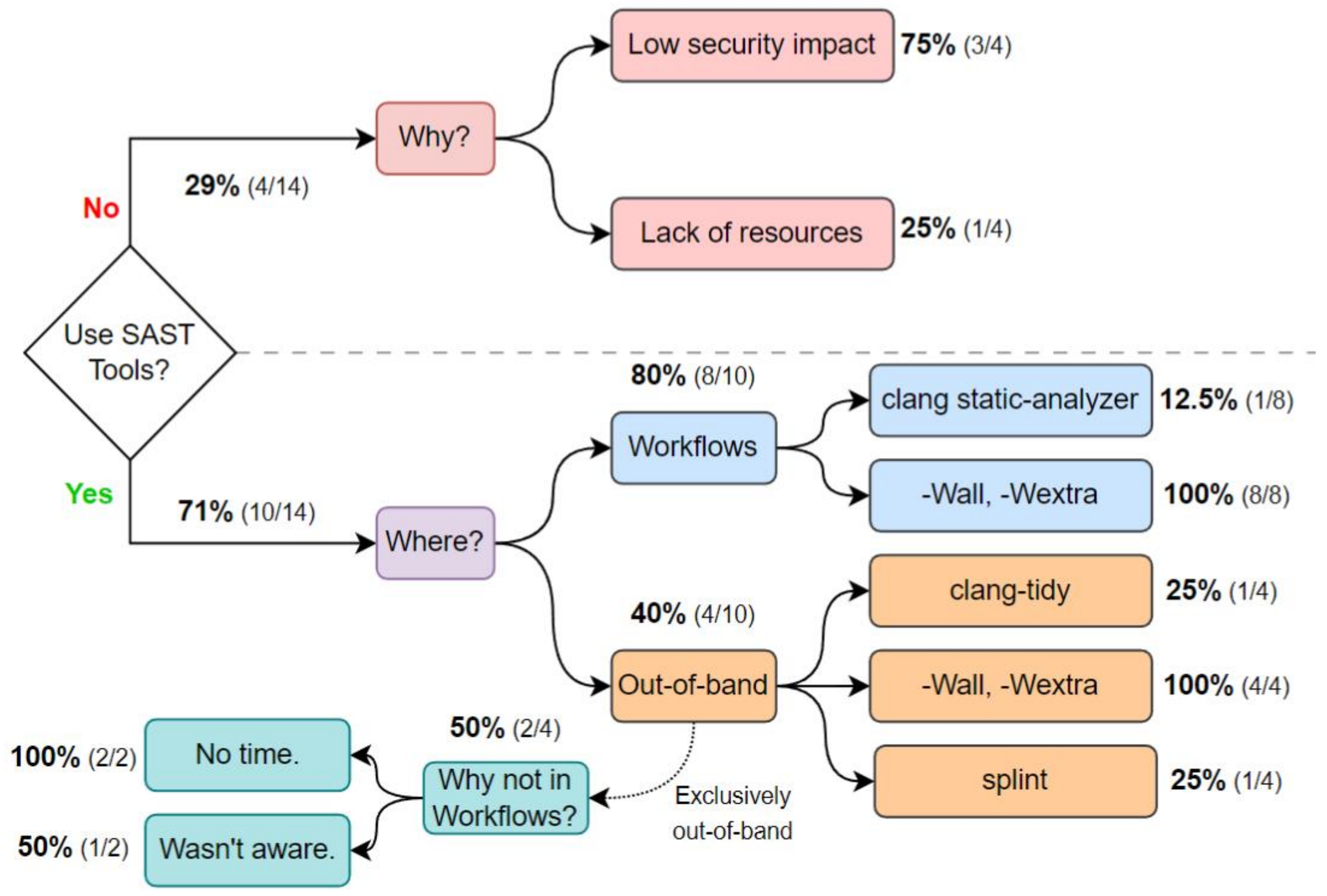
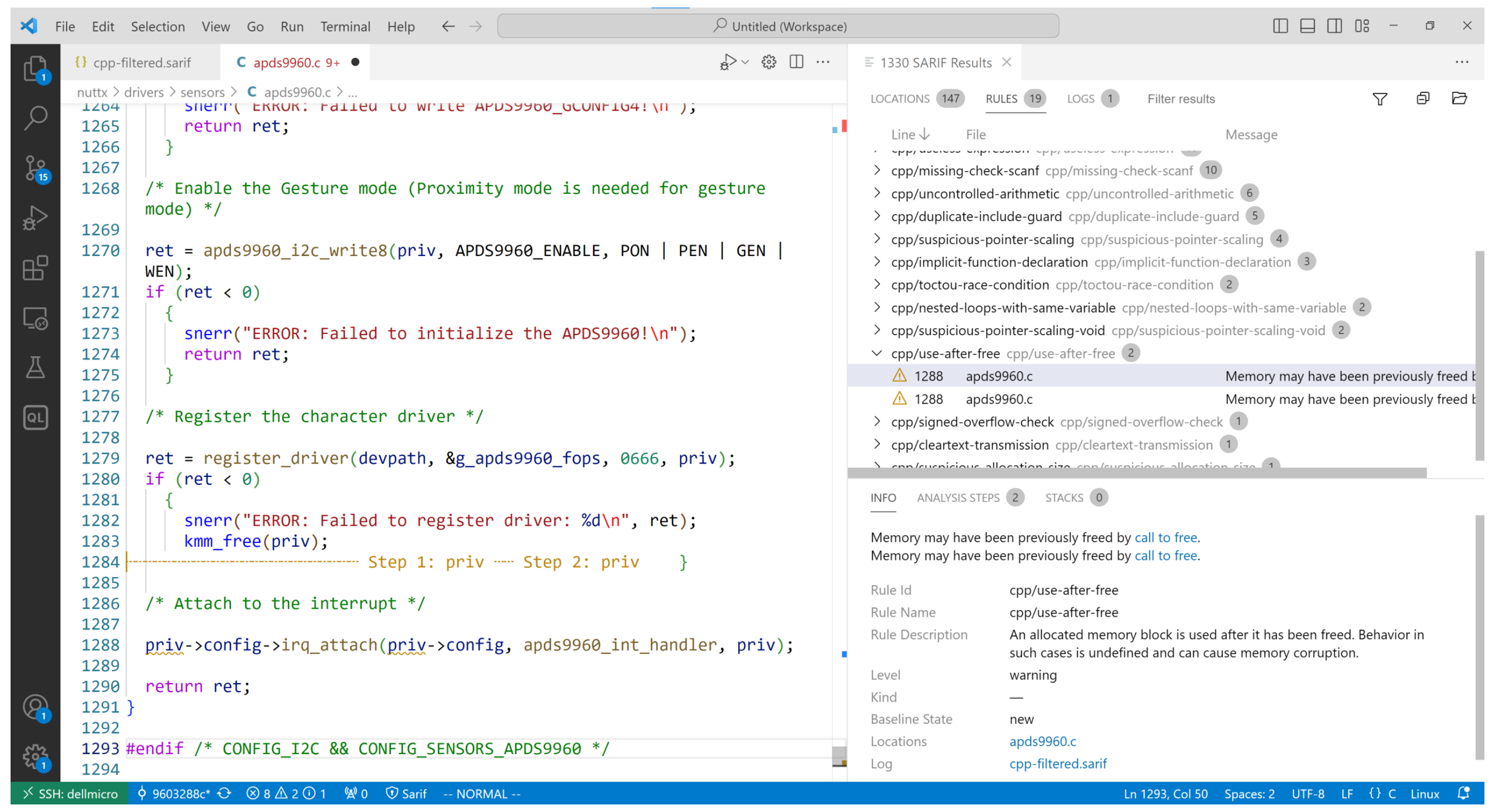


Fig: Summary of our developer survey on the use of SAST tools.

## RQ3: EFFECTIVENESS OF SAST TOOLS ON EMBOSS

- Getting CodeQL running takes minimal engineering effort, 45-60 min per project.
- CodeQL discovers many security and non-security defects
- Strict compiler warnings are less effective than CodeQL.
- The false positive rate (23%) of CodeQL meets developers' requirements.



```
      snerr("ERROR: Failed to write APDS9960_GCONFIG4!\n");
      return ret;
    }

  /* Enable the Gesture mode (Proximity mode is needed for gesture
  mode) */

  ret = apds9960_i2c_write8(priv, APDS9960_ENABLE, PON | PEN | GEN |
  WEN);
  if (ret < 0)
    {
      snerr("ERROR: Failed to initialize the APDS9960!\n");
      return ret;
    }

  /* Register the character driver */

  ret = register_driver(devpath, &g_apds9960_fops, 0666, priv);
  if (ret < 0)
    {
      snerr("ERROR: Failed to register driver: %d\n", ret);
      kmm_free(priv);
      ------------ Step 1: priv ---- Step 2: priv    }

  /* Attach to the interrupt */

  priv->config->irq_attach(priv->config, apds9960_int_handler, priv);

  return ret;
}

#endif /* CONFIG_I2C && CONFIG_SENSORS_APDS9960 */
```

```
> cpp/missing-check-scanf cpp/missing-check-scanf 10
> cpp/uncontrolled-arithmetic cpp/uncontrolled-arithmetic 6
> cpp/duplicate-include-guard cpp/duplicate-include-guard 5
> cpp/suspicious-pointer-scaling cpp/suspicious-pointer-scaling 4
> cpp/implicit-function-declaration cpp/implicit-function-declaration 3
> cpp/toctou-race-condition cpp/toctou-race-condition 2
> cpp/nested-loops-with-same-variable cpp/nested-loops-with-same-variable 2
> cpp/suspicious-pointer-scaling-void cpp/suspicious-pointer-scaling-void 2
v cpp/use-after-free cpp/use-after-free 2
    1288  apds9960.c   Memory may have been previously freed b
    1288  apds9960.c   Memory may have been previously freed b
> cpp/signed-overflow-check cpp/signed-overflow-check 1
> cpp/cleartext-transmission cpp/cleartext-transmission 1
```

Memory may have been previously freed by call to free.
Memory may have been previously freed by call to free.

| | |
|---|---|
| Rule Id | cpp/use-after-free |
| Rule Name | cpp/use-after-free |
| Rule Description | An allocated memory block is used after it has been freed. Behavior in such cases is undefined and can cause memory corruption. |
| Level | warning |
| Kind | — |
| Baseline State | new |
| Locations | apds9960.c |
| Log | cpp-filtered.sarif |

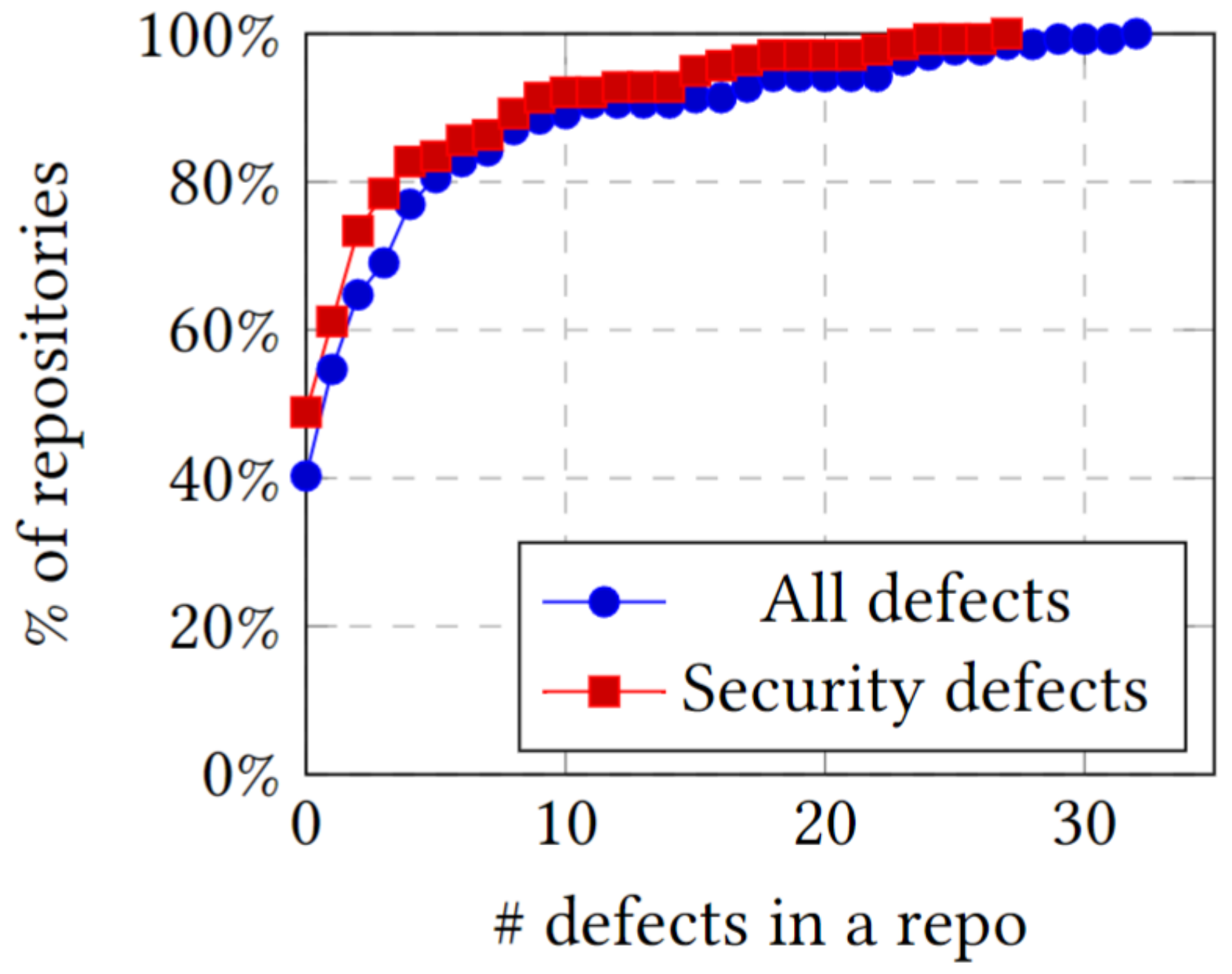


Fig: CDFs of # of total and security-relevant defects in a repository.

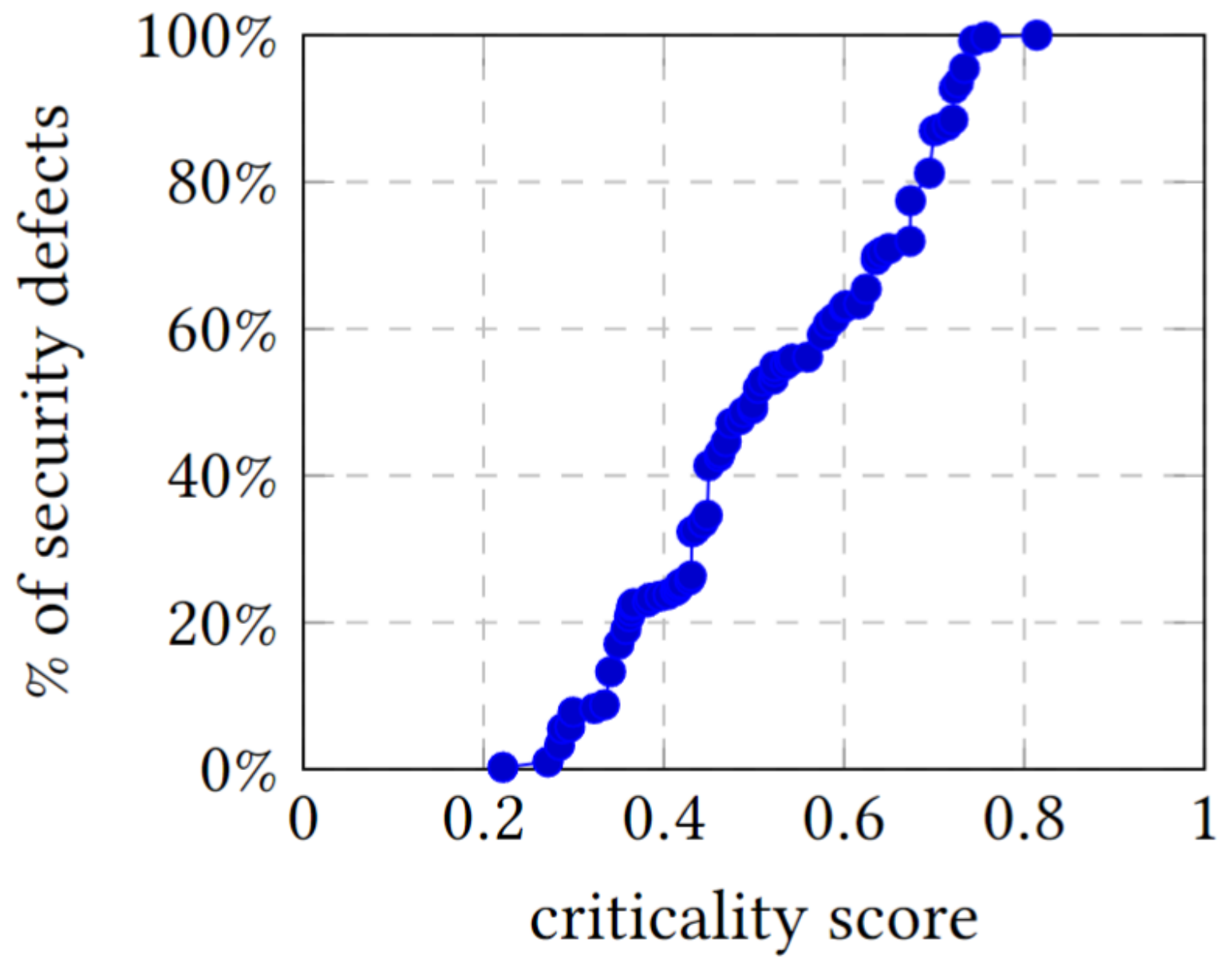


Fig: CDF of the severity of security defects.

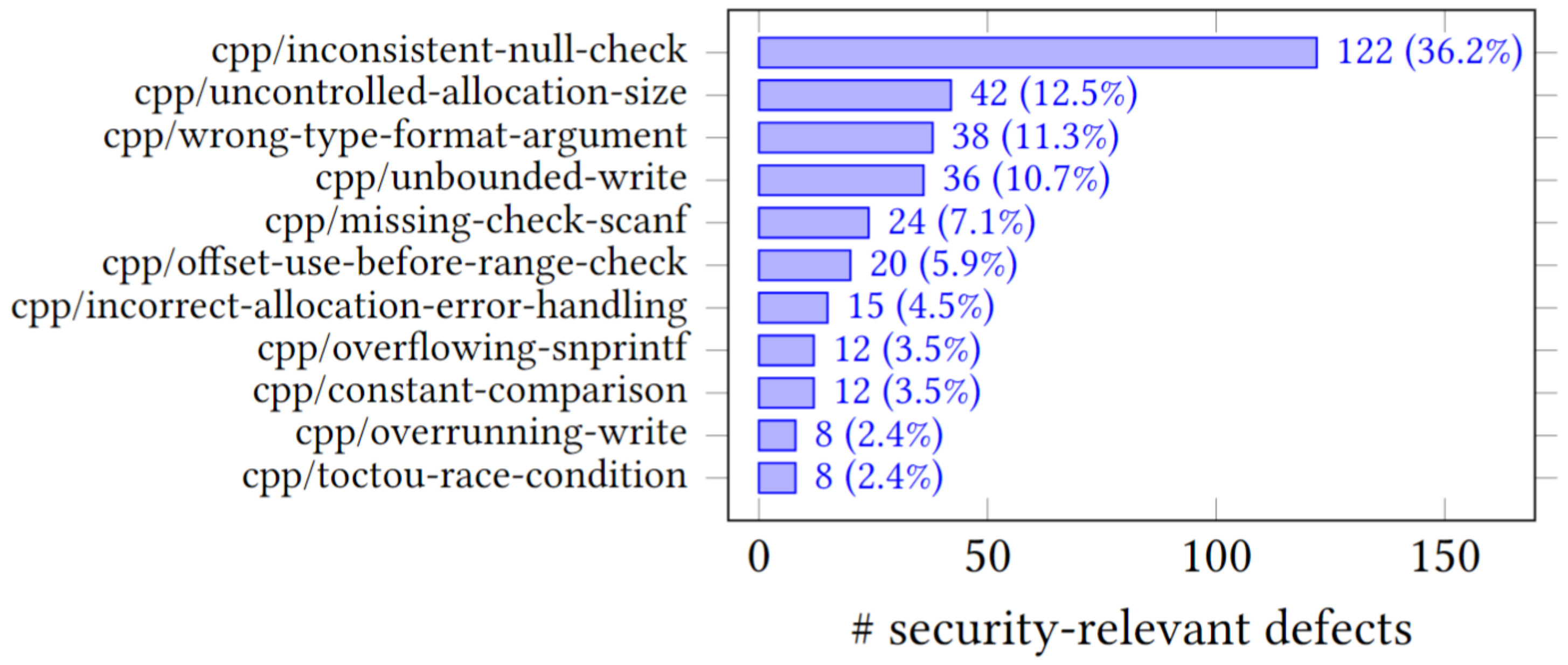


Fig: Top 10 CodeQL queries by # of security-relevant defects found.

Table: Results of SAST tools on EMBOSS repositories.

| Action | Result format | # Success Repo | # Failure Repo | Reasons for failure | Total # warn | Median # warn | Precision |
|---|---|---|---|---|---|---|---|
| david-a-wheeler/flawfinder | SARIF | 176 | 82 | Invalid SARIF, Python Error | 4,637 | 12 | 20% (64/316) |
| cpp-linter/cpp-linter-action | GCC error msg | 230 | 28 | Timeout, Python Error | 212,228 | 111 | 0% (0/213) |
| deep5050/cppcheck-action | GCC error msg | 256 | 2 | Timeout | 31,873 | 19 | 58% (116/200) |
| CodeQL Autobuild | SARIF | 74 | 184 | Autobuild failure | 471 | 0 | 96% (154/160) |

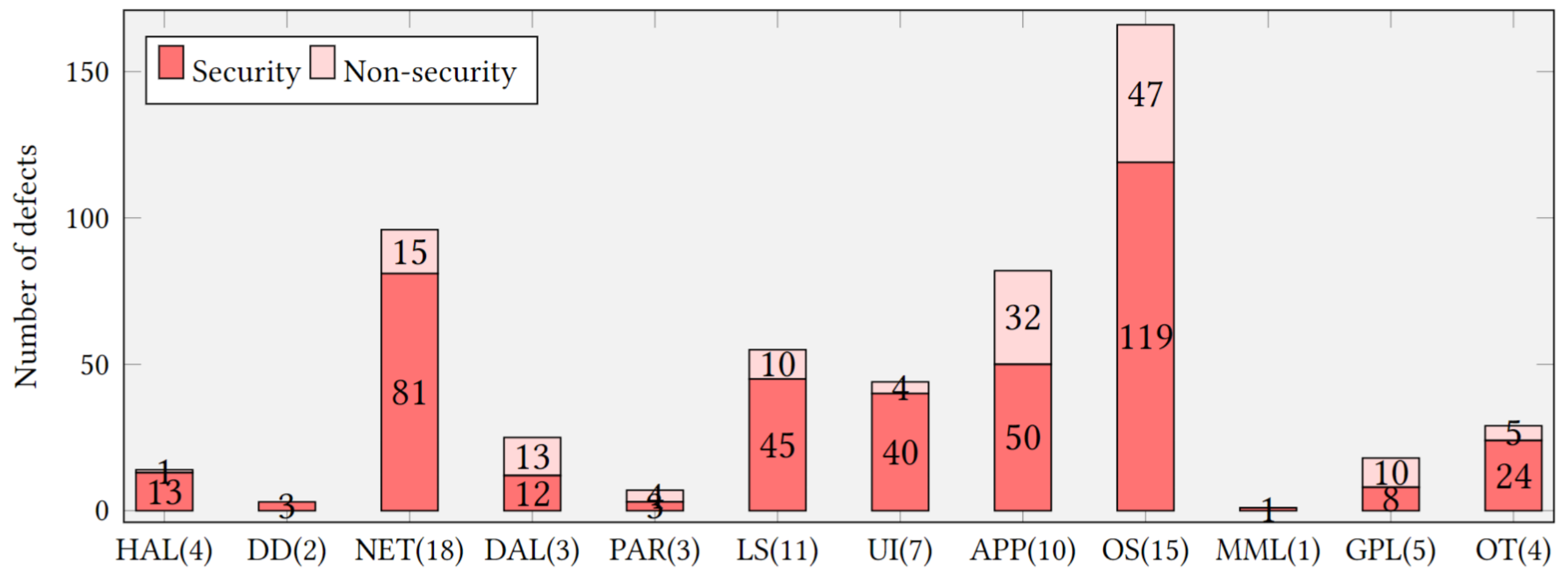


Fig: Number of defects of each type in EMBOSS of various categories. Category (#repo containing defects)

Table: Summary of CodeQL results and their analysis.

| Number of ... | Value |
|---|---|
| **Setup** | |
| Repos in dataset | 258 |
| Repos built | 154 |
| Repos analyzed | 143 |
| **CodeQL Results** | |
| Errors reported | 578 |
| Warnings reported | 2,294 |
| **Manual Analysis** | |
| Defects discovered | 540 |
| Repos where defects were discovered | 83 (60%) |
| Security defects discovered | 399 |
| Repos where security defects were discovered | 71 (51%) |
| **Responsible Disclosure** | |
| Defects confirmed | 273 |
| Security defects confirmed | 219 |
| Pull requests raised | 139 |
| Pull requests merged | 81 |
| CVEs issued | 2 |

## RQ2: CHALLENGES IN EFFECTIVELY USING SAST TOOLS

- Warnings produced in a non-standard text format
- CodeQL autobuild fails to handle the diverse build infrastructure of the majority repositories
- Preliminary evaluation shows that CodeQL has the highest precision on EMBOSS repositories.