

Triaging CVEs for the Linux Kernel

Christoph Steiger
Negar Masoudifar

About Us



Christoph Steiger

- Software Engineer @Siemens Linux Expert Center



@Urist-McGit



christoph.steiger@siemens.com



Negar Masoudifar

- Working Student



@n.masoudifar



negar.masoudifar@siemens.com

Civil Infrastructure Platform (CIP)

Version	Maintainer(s)	First Release	Projected <u>EOL</u>	Target Releases/Month*
SLTS v6.12	Nobuhiro Iwamatsu & Pavel Machek	2025-05-20	2035-06	2
SLTS v6.1	Nobuhiro Iwamatsu & Pavel Machek	2023-07-14	2033-08	1
SLTS v6.1-rt	Pavel Machek	2023-07-16	2033-08	0.5
SLTS v5.10	Nobuhiro Iwamatsu & Pavel Machek	2021-12-05	2031-01	1
SLTS v5.10-rt	Pavel Machek	2021-12-08	2031-01	0.5
SLTS v4.19	Ulrich Hecht	2019-01-11	2029-01	1
SLTS v4.19-rt	Pavel Machek	2019-01-11	2029-01	0.5
SLTS v4.4	Ulrich Hecht	2017-01-17	2027-01	1
SLTS v4.4-rt	Pavel Machek	2017-11-16	2027-01	0.5

CIP; Licensed under [CC-BY-NC-SA 4.0](#)

- Project is part of the Linux Foundation
- Aims to provide a "base layer" for industrial grade software
- Specifically: [linux-cip](#) kernel with SLTS support
- Development of kernel-cve-triage tool for vulnerability assessment

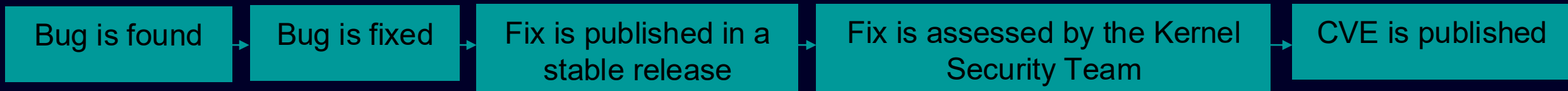


Agenda

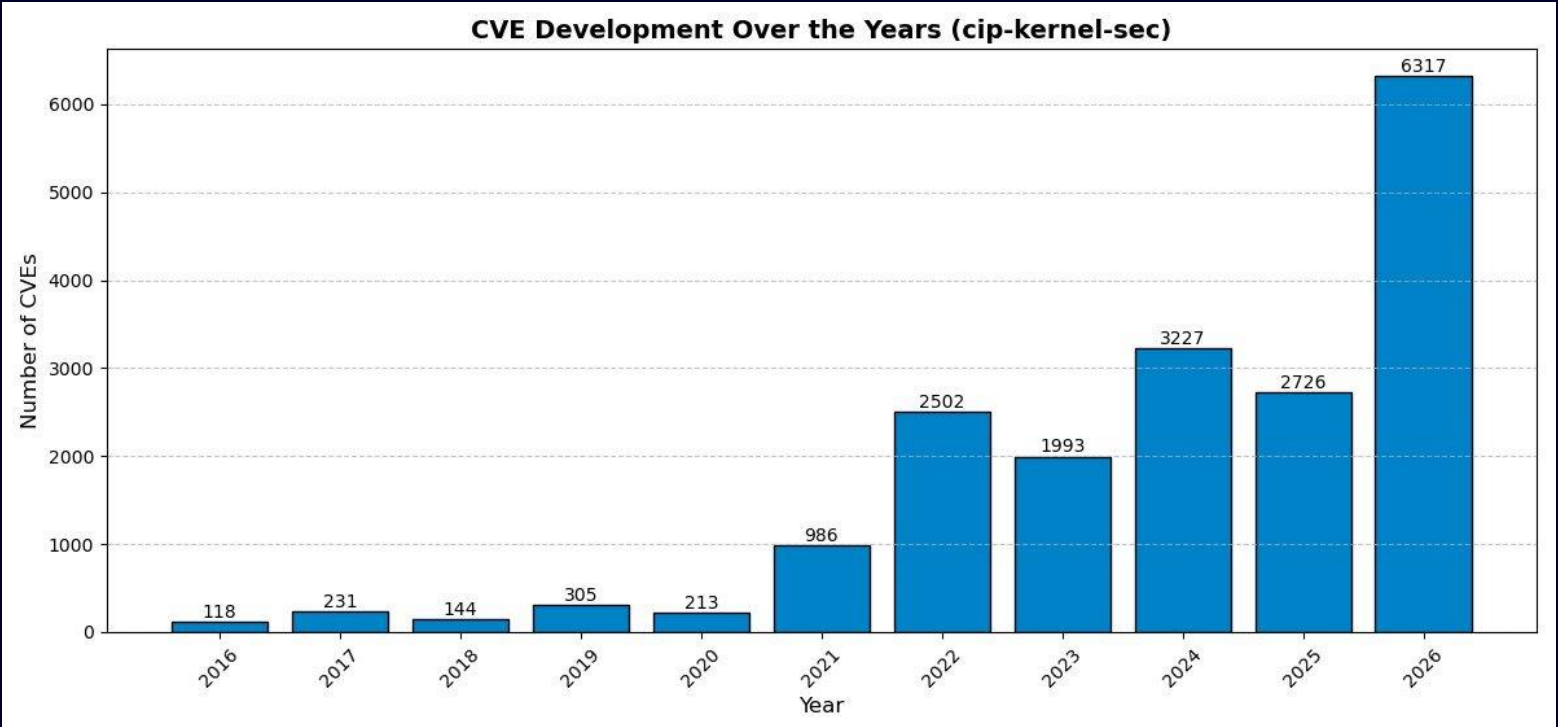
1. Kernel Vulnerability Handling Overview
2. Approaches for Kernel Vulnerability Remediation
3. Implementation Overview for kernel-cve-triage and Yocto CVE Check
4. Results & Comparison
5. Summary & Outlook

Overview: Vulnerabilities in the Kernel

- See [summary from GKH](#) about the topic for more in-depth look
- Kernel Team started issuing CVEs as CNA in February 2024
- In 2025: ~60 CVEs per week; in 2026: ~150 CVEs per week
- Kernel is not insecure, it is just very big and *very conservative* in its vulnerability assessment
- CVE workflow is special in the Kernel: fixes come into stable releases and are then checked for their security impact

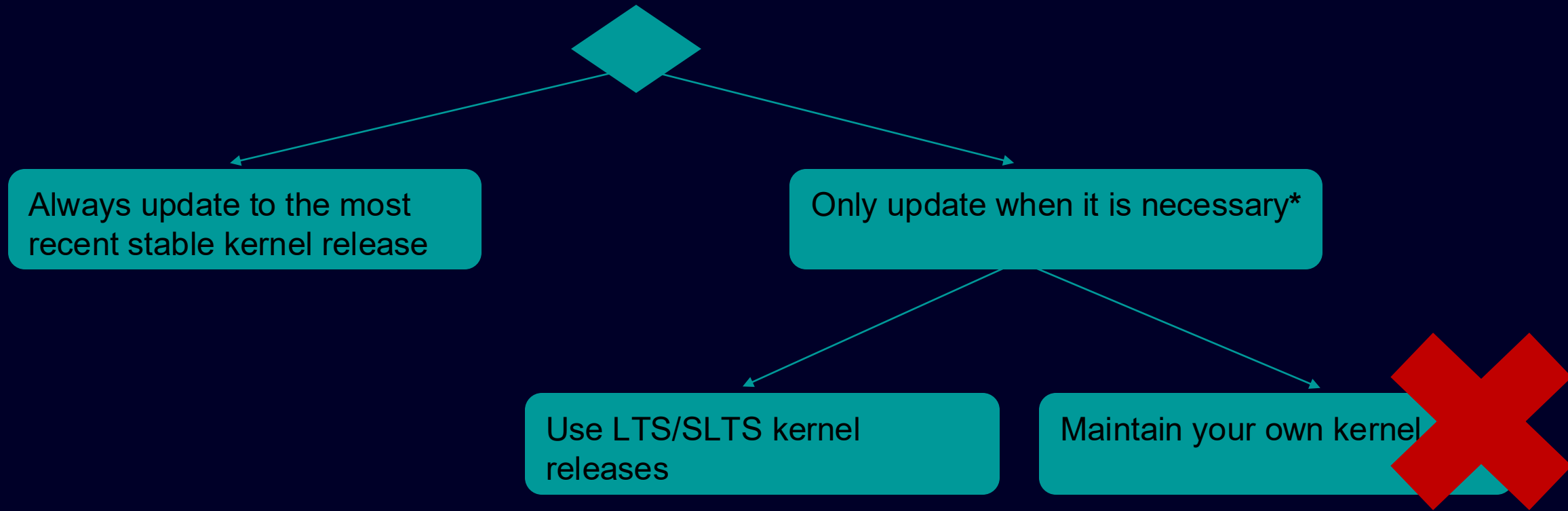


Overview: Vulnerabilities in the Linux Kernel



CRA imposes compliance requirements: it is no longer enough to give a blanket statement about kernel security

Approaches for Vulnerability Remediation



Deciding when it is necessary to update is the hard part

Implementation Overview: kernel-cve-triage

Starting point: CVE, kernel source tree and version and a config

1: Check if our version might be affected

2: Based on the CVE commit information find affected files

3: Find affected configuration options with kmax

4: Check if affected configuration option is in kernel config

Usage Examples

Create a report:

```
cve-triage triage -c .config -s linux-cip:v6.12.96-cip28 issues/CVE-2026-*
```

(triates the linux-cip v6.12.96-cip28 kernel with the given .config and all 2026 CVEs)

Annotate CVEs with affected configuration options and files:

```
cve-triage enrich -s linux-cip --versions v6.12 -- issues/CVE-2026-*
```

Usage Examples

```
{
  "check_results": [
    {
      "name": "check_fixed_by",
      "result": "not relevant",
      "reason": "found ['8f34f1eac3820fc2722e5159accebb22545b30b0d']"
    },
    {
      "name": "check_ignore",
      "result": "relevant",
      "reason": "no ignore"
    },
    {
      "name": "check_introduced_by",
      "result": "relevant",
      "reason": "found b569a1760782f3da03ff718d61f74163dea599ff"
    },
    {
      "name": "check_introduced_by_file_compiled",
      "objects": [
        {
          "name": "mm/memory.o",
          "built": true,
          "config": "CONFIG_MMU"
        },
        {
          "name": "mm/huge_memory.o",
          "built": true,
          "config": "CONFIG_TRANSPARENT_HUGEPAGE"
        }
      ],
      "deleted_files": [],
      "result": "relevant",
      "reason": "at least one file is compiled"
    },
    {
      "name": "check_fixed_by_file_compiled",
      "objects": [],
      "added_later": [],
      "result": "relevant",
      "reason": "header search depth exceeded, too many files to analyze"
    }
  ]
},
{
```

```
{
  "check_results": [
    {
      "name": "check_fixed_by",
      "result": "relevant",
      "reason": "fixed-by not found"
    },
    {
      "name": "check_ignore",
      "result": "relevant",
      "reason": "no ignore"
    },
    {
      "name": "check_introduced_by",
      "result": "relevant",
      "reason": "found 9827a0e6e23bf43003cd3d5b7fb11baf59a35e1e"
    },
    {
      "name": "check_introduced_by_file_compiled",
      "objects": [
        {
          "name": "net/netfilter/nft_set_pipapo.o",
          "built": true,
          "config": "CONFIG_NF_TABLES && CONFIG_NETFILTER"
        }
      ],
      "deleted_files": [],
      "result": "relevant",
      "reason": "at least one file is compiled"
    },
    {
      "name": "check_fixed_by_file_compiled",
      "objects": [
        {
          "name": "net/netfilter/nft_set_pipapo.o",
          "built": true,
          "config": "CONFIG_NF_TABLES && CONFIG_NETFILTER"
        }
      ],
      "added_later": [],
      "result": "relevant",
      "reason": "at least one file is compiled"
    }
  ]
},
{
```

Implementation Overview & Comparison

kernel-cve-triage

Starting point: CVE, kernel source tree with git history, version and a config

1: Check if our version might be affected

2: Based on the CVE commit information find affected files

3: Find affected configuration options with kmax

4: Check if affected configuration option is in kernel config

Yocto CVE Check

Starting point: CVE (with affected file info), yocto build system

1: Check if our version might be affected

2: Extract included files from debuginfo during build

3: Check if affected files are built

Tool Differences

kernel-cve-triage

Yocto CVE Check

Inputs	source tree with git history	kernel build artifact
Affected file information	automatically derived from CVE commit information	directly taken from kernel CVEs
Assessment Criteria	Kconfig configuration options	files

Comparing kernel-cve-triage with Yocto CVE Check

Use similar Inputs

Use the same
kernel version and
identical config file



Run both tools independently

kernel-cve-triage

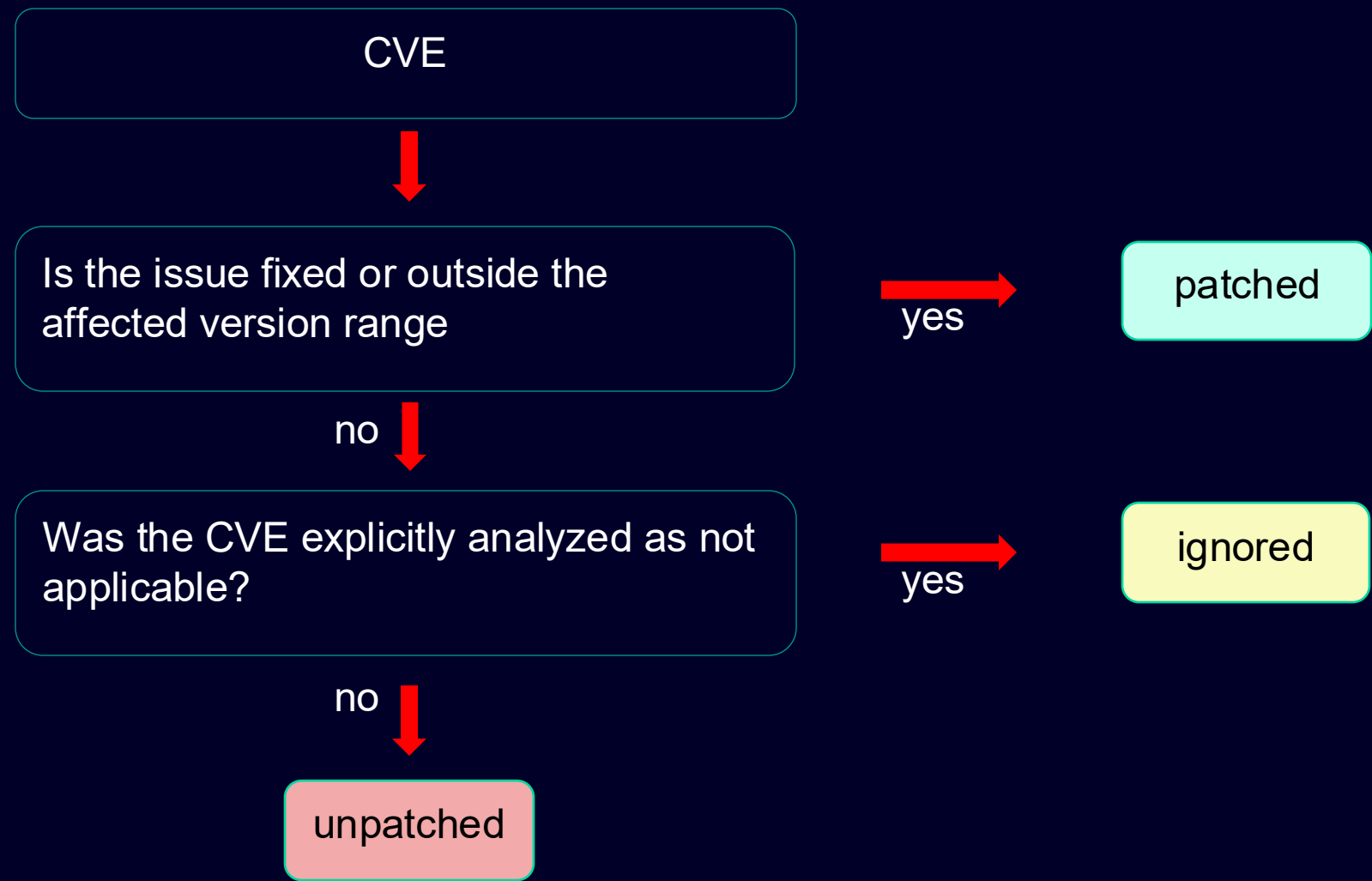
Yocto cve-check



Compare outputs

Check for identical
cves there is same
conclusion

Comparing kernel-cve-triage with Yocto CVE Check



Comparing kernel-cve-triage with Yocto CVE Check

CVE Selection Criteria:

1 month before release:

captures vulnerabilities that were known during the final development and stabilization phase and are likely relevant to the released version.

Up to 6 months after release:

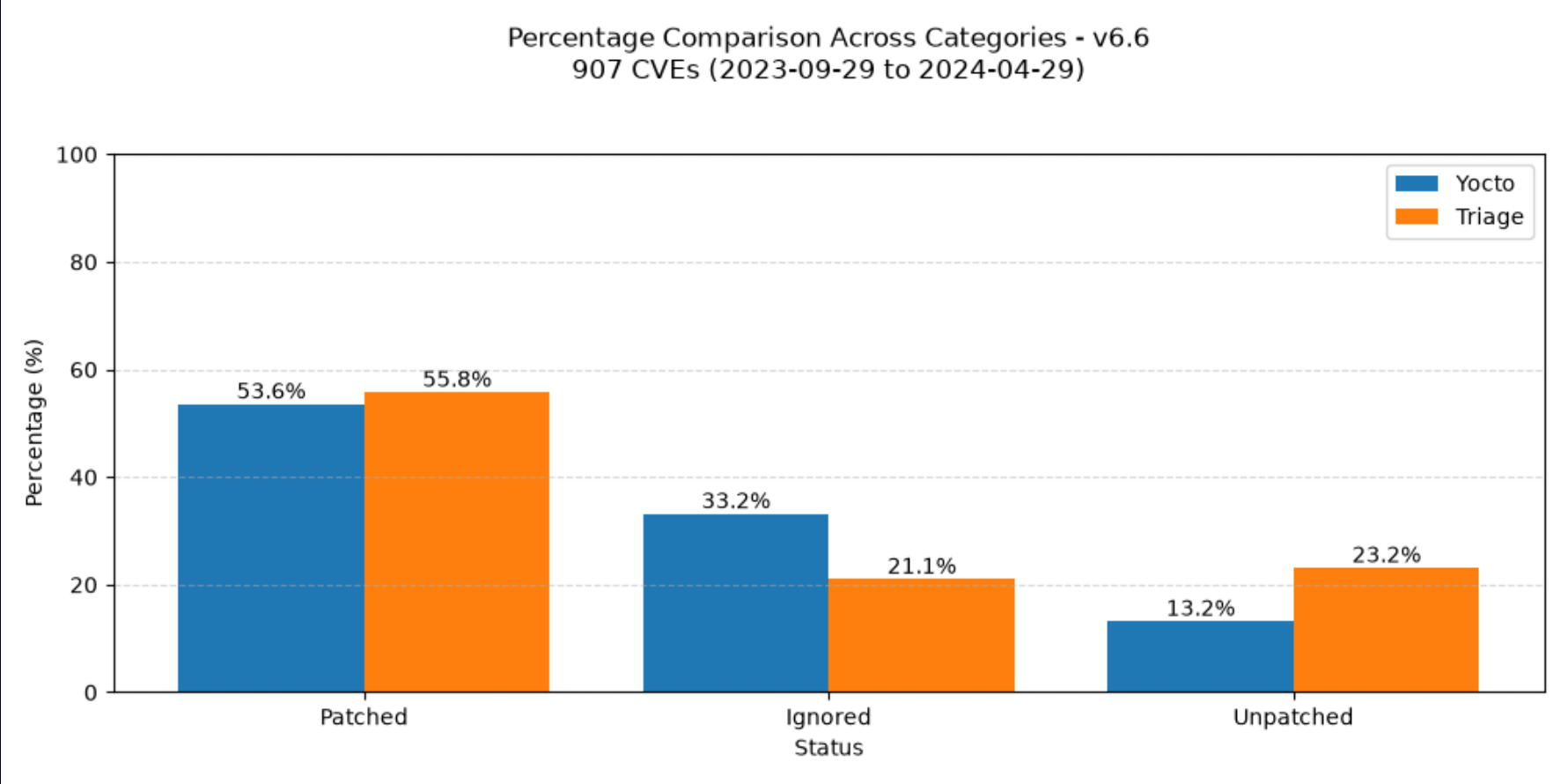
captures vulnerabilities affecting the version that were discovered and disclosed shortly after deployment, providing a realistic view of its security exposure.

Comparing kernel-cve-triage with Yocto CVE Check

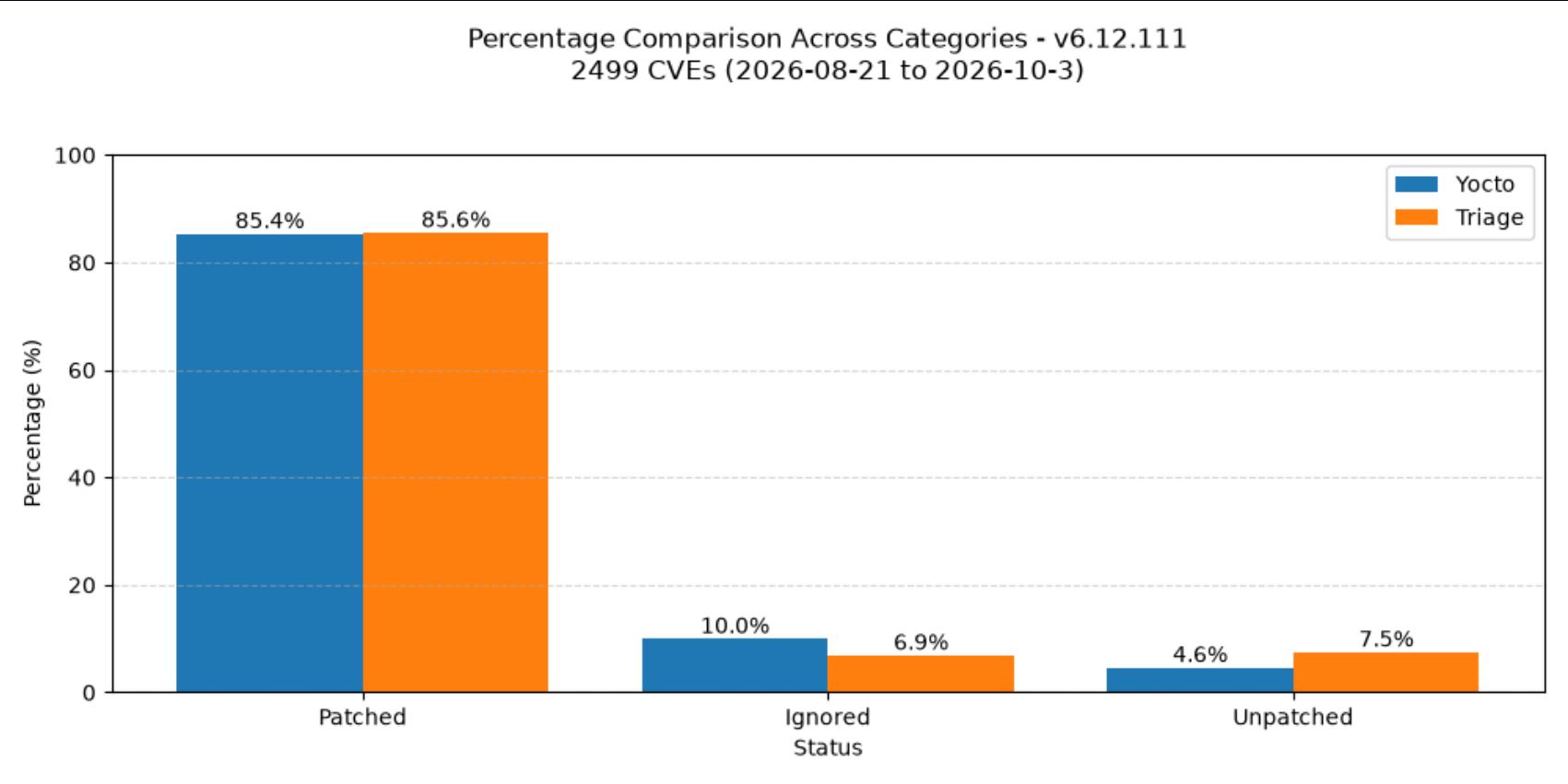
Key Findings:

- The **patched** and **unpatched** classifications show a high level of agreement between the kernel-cve-triage and Yocto CVE Check.
- Most differences are observed in the ignored category.
- One reason is the dependency on Kmax for configuration analysis:
 - In some cases, Kmax fails to identify the relevant kernel configuration.
 - File determination can fail

Comparing kernel-cve-triage with Yocto CVE Check



Comparing kernel-cve-triage with Yocto CVE Check



Summary & Outlook

- Current tools can realistically filter ~80% of CVEs, further improvements have to come from different approaches
- Generic description of threat models and attack vectors might be able to improve current automated assessments
- Workload can be shared: use LTS and SLTS offerings; join communities like CIP
- Suggestion: include affected Kconfig configuration options in kernel CVEs

Thank you for your attention!

CIP Project: <https://cip-project.org/>

kernel-cve-triage: <https://gitlab.com/cip-project/cip-kernel/kernel-cve-triage>

Yocto CVE Check: <https://docs.yoctoproject.org/next/security-manual/vulnerabilities.html#linux-kernel-vulnerabilities>

Questions?

