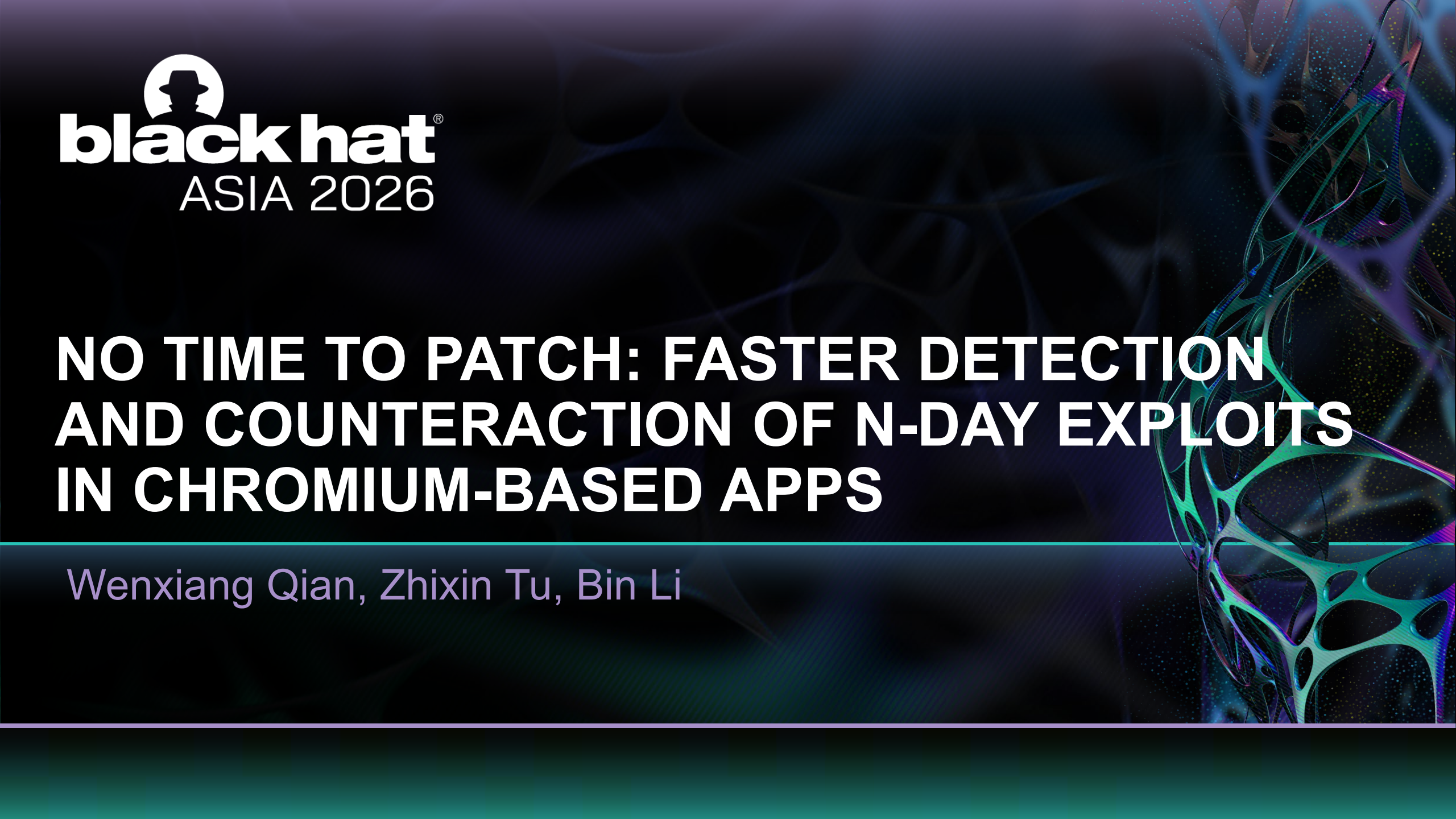

black hat®
ASIA 2026
APRIL 21-24, 2026
MARINA BAY SANDS / SINGAPORE



NO TIME TO PATCH: FASTER DETECTION AND COUNTERACTION OF N-DAY EXPLOITS IN CHROMIUM-BASED APPS

Wenxiang Qian, Zhixin Tu, Bin Li

ABOUT US

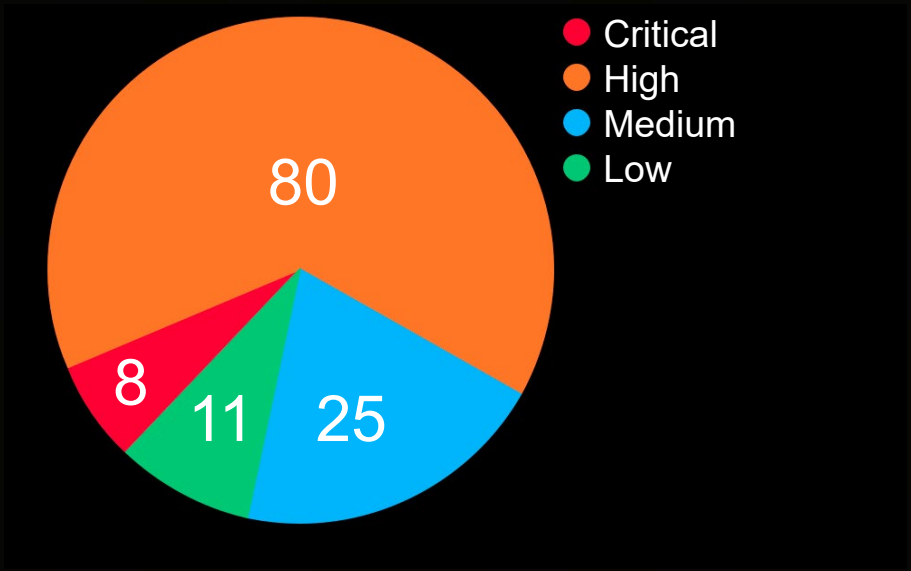
- **Wenxiang Qian (@qian_wenxiang)**
  - Security researcher and engineer specializing in application and browser security architecture. Black Hat Speaker (USA 2019, 2021). Book author.
- **Zhixin Tu (@tu_zhixin)**
  - Security engineer specializing in browser security. Works on attack surface analysis and defensive hardening for client applications.
- **Bin Li (@bingoli20000)**
  - Senior expert in browser and client technologies. Specializing in software performance, development quality, and security research.

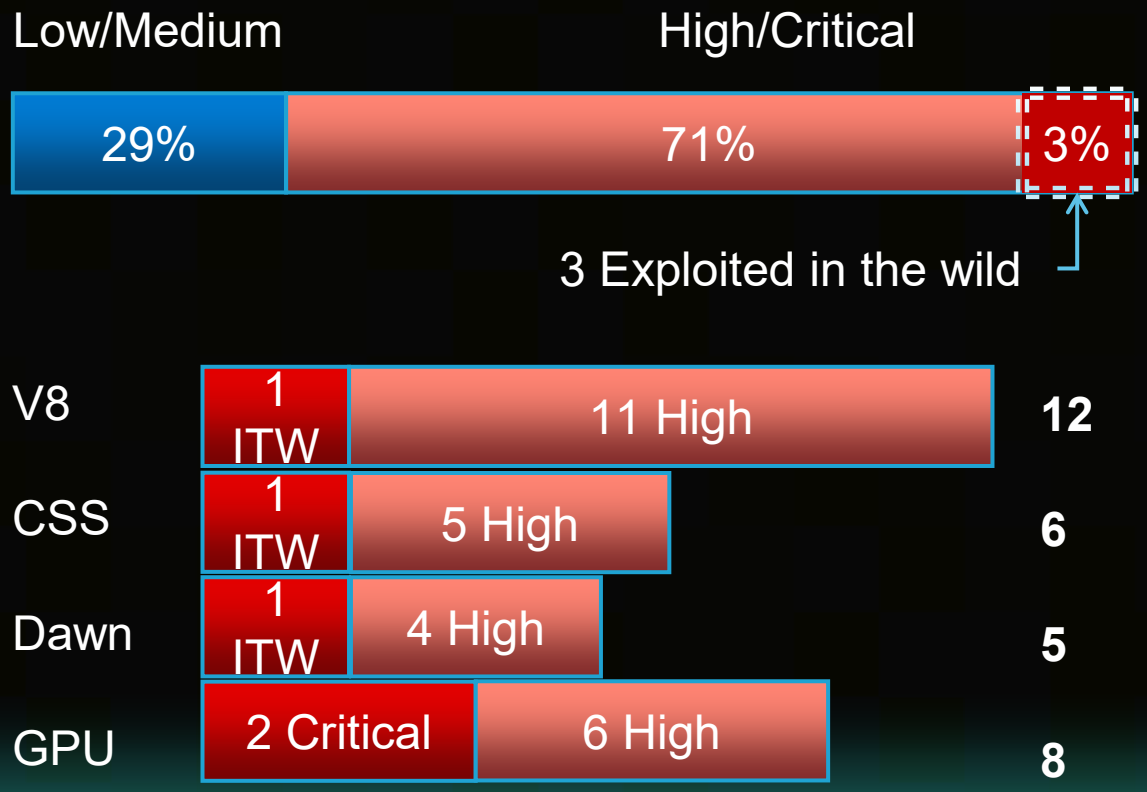
AGENDA

- Modern N-day Problems
- Automatic CVE Signature Analysis
- Automatic Defensive Strategy Analysis
- Applying Performance-Friendly Defensive Strategies
- Takeaways and Best Practices

STATISTICS FROM JAN TO MAR 2026

- In 2026Q1, Chrome has fixed 124 vulns with CVE





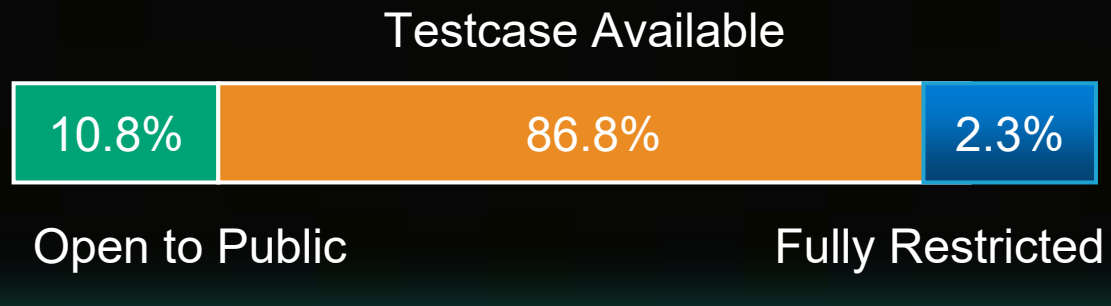
Scale may be adjusted for better illustration.

N-DAYS ARE MAIN THREAT

LOW COST

- Google releases after 14 weeks

Vulnerabilities (01 JAN - 17 MAR)
Status as of 02 April, 2026:



WIDE IMPACT

- Chromium are omnipresent



Scale may be adjusted for better illustration.

LET'S START WITH A STORY

05:06 AM

- 1st July 2025, Beijing Time 05:06 AM
  - After 4 hours, a massive Red Team match will be started nation-wide, we are the blue team (defender team)
- Our annoying robot alerts in-the-wild CVE-2025-6554



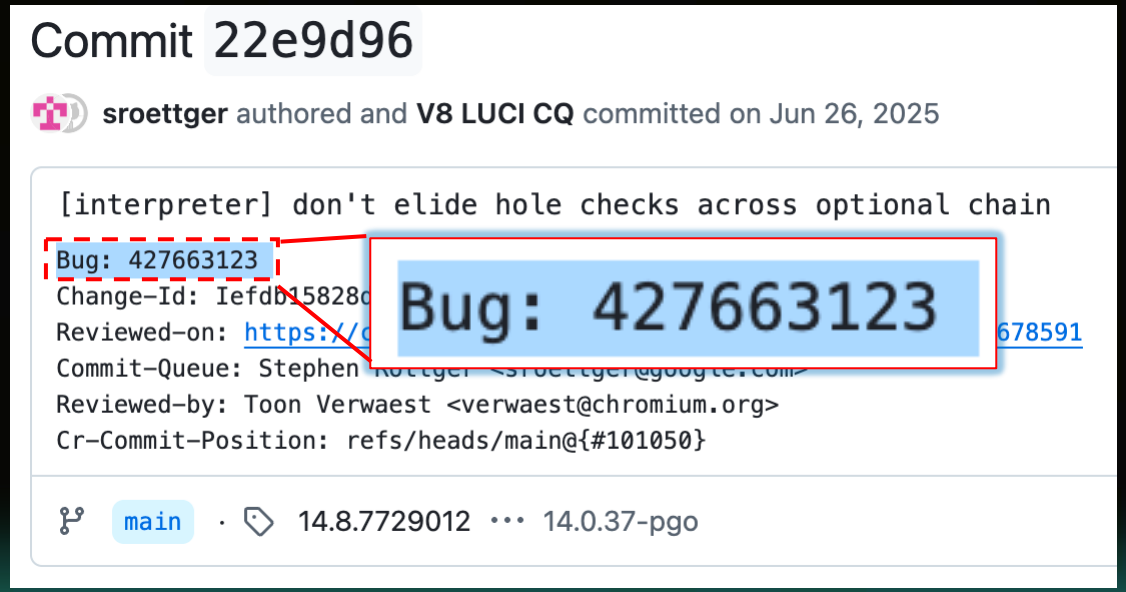
- But all of us are sleeping right now
- How do we respond?

THE RACING WITH THE ATTACKER STARTS

- Patch obtained by BugID: 427663123

05:30 AM

1



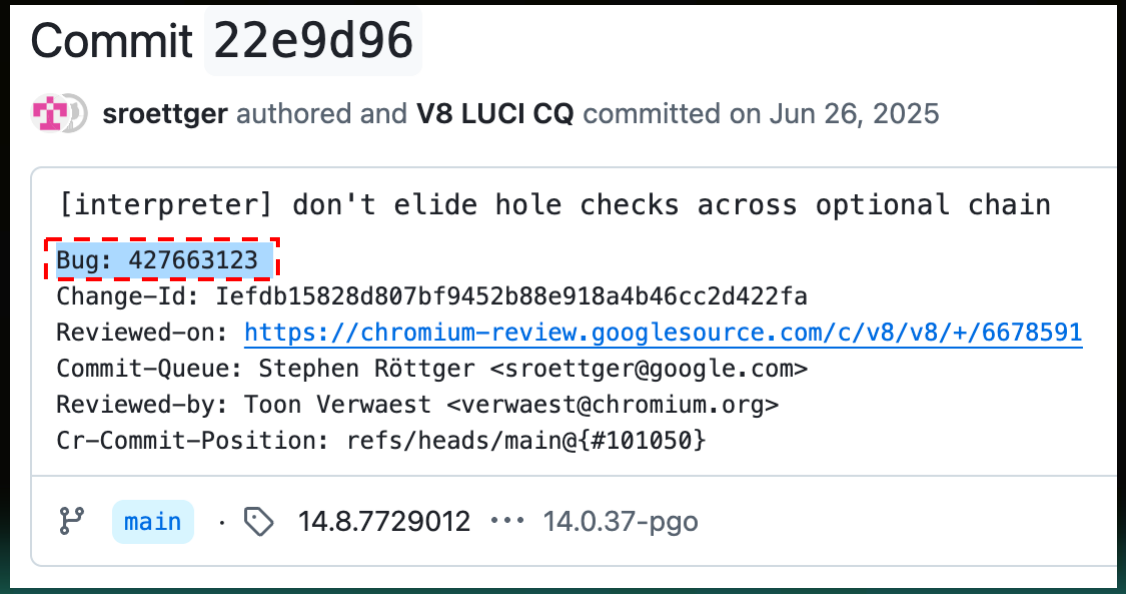
black hat ASIA 2026

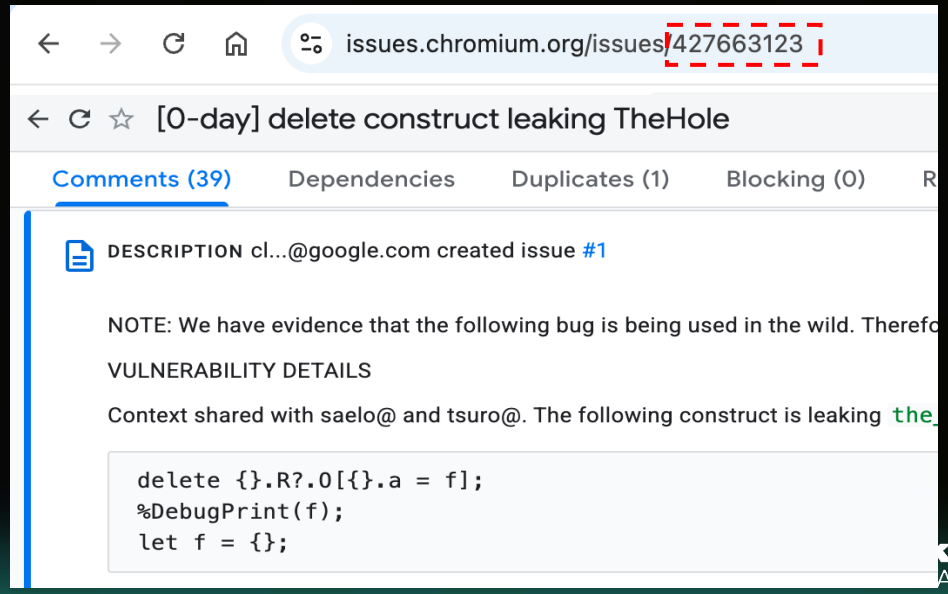
THE RACING WITH THE ATTACKER STARTS

05:30 AM

1

- Patch obtained by BugID: 427663123
- PoC obtained (Google Developer Only)



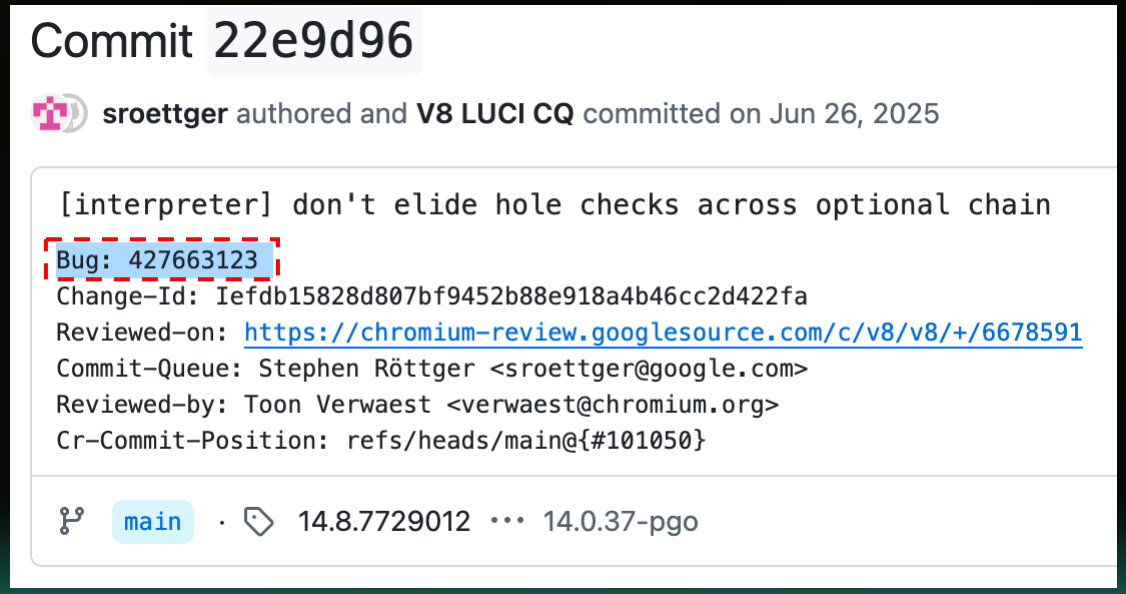


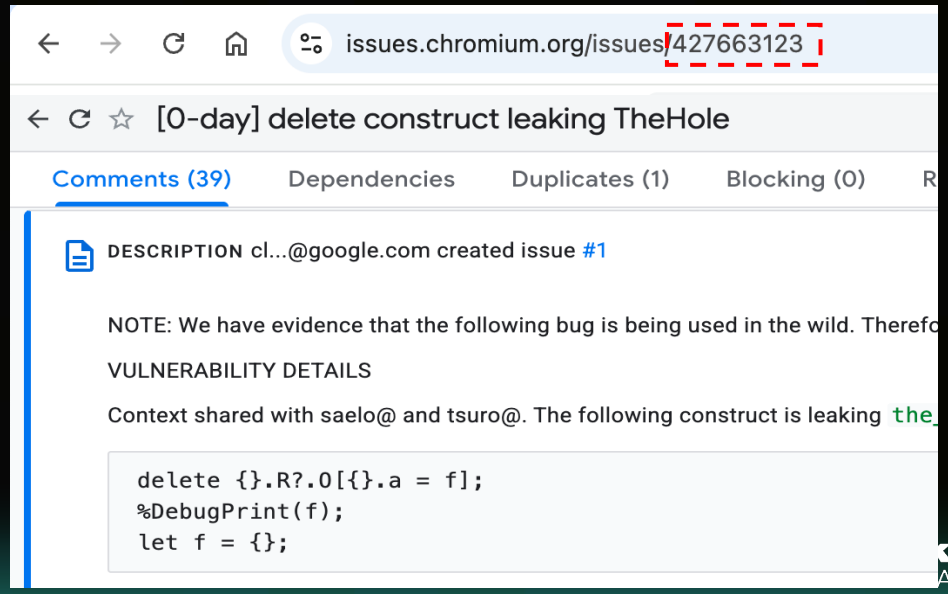
THE RACING WITH THE ATTACKER STARTS

05:30 AM

1

- Patch obtained by BugID: 427663123
- PoC obtained (Google Developer Only)
- Test PoC through precompiled M109 to M138





THE RACING WITH THE ATTACKER STARTS

06:00 AM

- Perform RCA

2

CVE-2025-6554 Type Confusion in V8.

Utilization in the wild	yes
bonus	10000
Number of patches	3
First patch	2023-05-22T15:47:46
Latest Patch	2025-06-26T09:13:24
Intelligence source	Clment Lecigne of Google's Threat Analysis Group
weaponization probability	100.00%
Affected components	v8
Affected versions	m115–m140
Vulnerability Principles	The root cause of the vulnerability lies in a flawed combination of the V8 interpreter bytecode generator's omission of TDZ (Temporal Dead Zone) "hole" checks and short-circuiting of optional chains. Specifically, the PoC construct `delete {}.R?.O[{}.a = f]; %DebugP`

THE RACING WITH THE ATTACKER STARTS

06:00 AM

2

- Perform RCA
- Patching our codebase, testing against unit-tests

CVE-2025-6554 Type Confusion in V8.

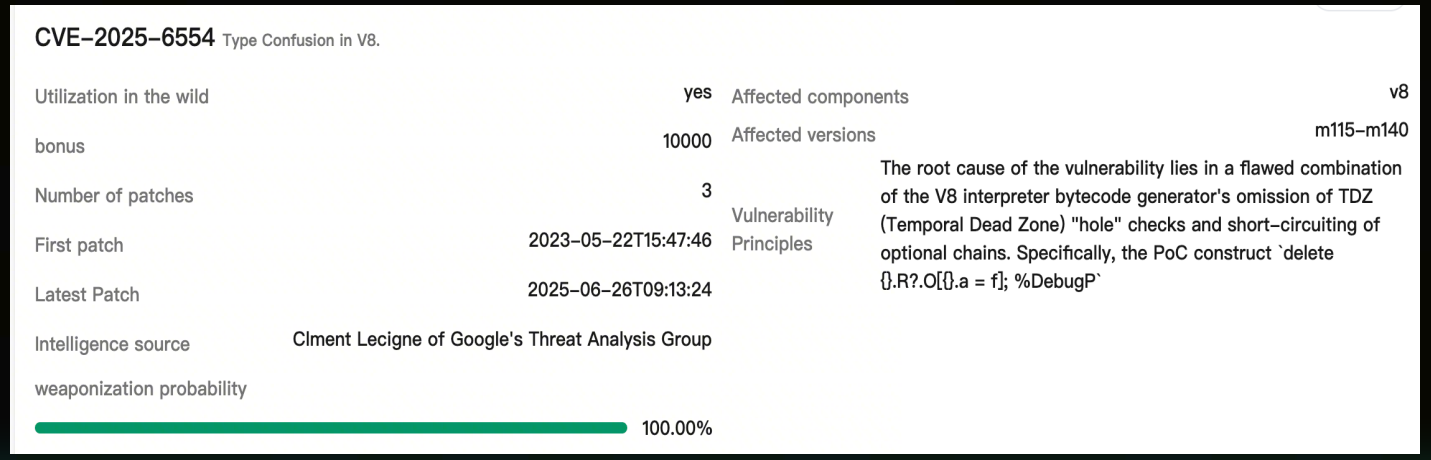
Utilization in the wild	yes
bonus	10000
Number of patches	3
First patch	2023-05-22T15:47:46
Latest Patch	2025-06-26T09:13:24
Intelligence source	Clment Lecigne of Google's Threat Analysis Group
weaponization probability	100.00%
Affected components	v8
Affected versions	m115–m140
Vulnerability Principles	The root cause of the vulnerability lies in a flawed combination of the V8 interpreter bytecode generator's omission of TDZ (Temporal Dead Zone) "hole" checks and short-circuiting of optional chains. Specifically, the PoC construct `delete {}.R?.O[{}.a = f]; %DebugP`

THE RACING WITH THE ATTACKER STARTS

06:00 AM

2

- Perform RCA
- Patching our codebase, testing against unit-tests
- Set crash & social media monitoring

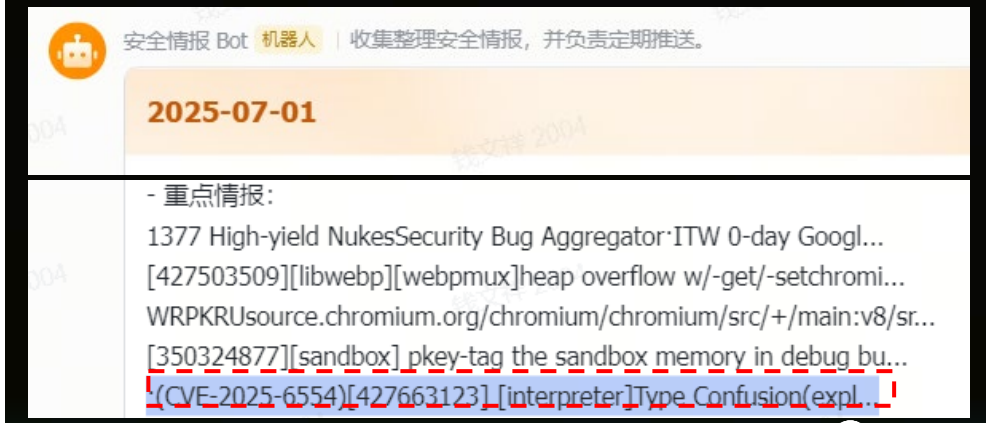


CVE-2025-6554 Type Confusion in V8.

Utilization in the wild	yes	Affected components	v8
bonus	10000	Affected versions	m115–m140
Number of patches	3	Vulnerability Principles	The root cause of the vulnerability lies in a flawed combination of the V8 interpreter bytecode generator's omission of TDZ (Temporal Dead Zone) "hole" checks and short-circuiting of optional chains. Specifically, the PoC construct `delete {}.R?.O[{}.a = f]; %DebugP`
First patch	2023-05-22T15:47:46		
Latest Patch	2025-06-26T09:13:24		
Intelligence source	Clment Lecigne of Google's Threat Analysis Group		

weaponization probability

100.00%



安全情报 Bot 机器人 | 收集整理安全情报，并负责定期推送。

2025-07-01

- 重点情报:

1377 High-yield NukesSecurity Bug Aggregator·ITW 0-day Googl...

[427503509][libwebp][webpmux]heap overflow w/-get/-setchromi...

WRPKRUsource.chromium.org/chromium/chromium/src/+/main:v8/sr...

[350324877][sandbox] pkey-tag the sandbox memory in debug bu...

(CVE-2025-6554)[427663123] [interpreter]Type Confusion(expl...

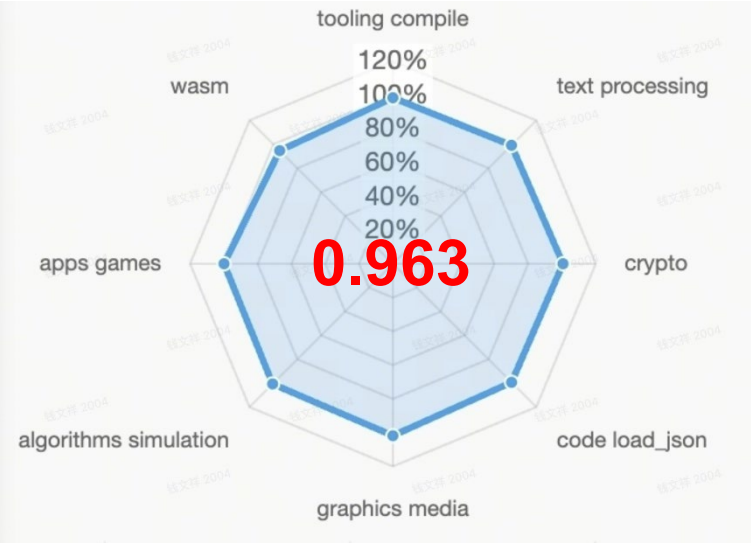
THE RACING WITH THE ATTACKER STARTS

- 7 JSFlags proposed, best selected

08:00 AM

3

--no-ignition_elide_redundant_tdz_checks



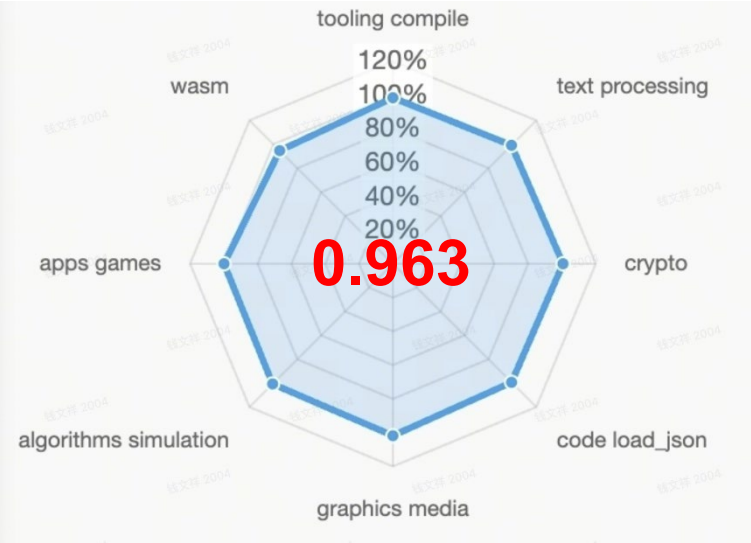
THE RACING WITH THE ATTACKER STARTS

08:00 AM

3

- 7 JSFlags proposed, best selected
- Test against every version of our App
- Failed test waiting for human specialist to check

--no-ignition_elide_redundant_tdz_checks




```
[ RUN      ] BytecodeGeneratorTest.Modules
Inputs differ at line 76
  Generated: 'bytecode array length: 69'
  Expected:  'bytecode array length: 67'
Inputs differ at line 101
  Generated: 'B(ThrowReferenceErrorIfHole), U8(3),'
  Expected:  'B(Star3),'
Inputs differ at line 102
  Generated: 'B(Star3),'
  Expected:  'B(LdaSmi), I8(42),'
Inputs differ at line 103
  Generated: 'B(LdaSmi), I8(42),'
  Expected:  'B(Star4),'
```

Added bytecode caused misalign

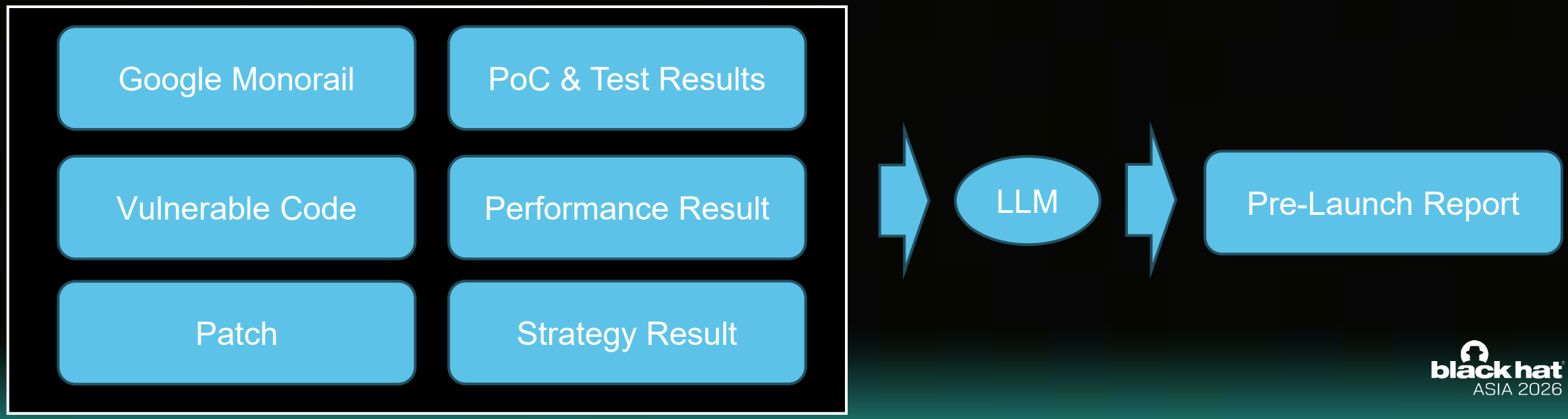
black hat ASIA 2026

THE RACING WITH THE ATTACKER STARTS

09:00 AM

4

- Performance and stability test finished
- Specialist approves the strategy
- QA Passed, greyscale in progress



black hat ASIA 2026

THE RACING WITH THE ATTACKER STARTS

- Defensive JSFlag protects our customer

11:45 AM

5

THE RACING WITH THE ATTACKER STARTS

11:45 AM

5

- Defensive JSFlag protects our customer
- PoC Leaked in the afternoon (03:27PM)
- First work exploit released next day (04:30PM)

Leak hole PoC for Chrome in-the-wild vulnerability CVE-2025-6554 published yesterday:
github.com/

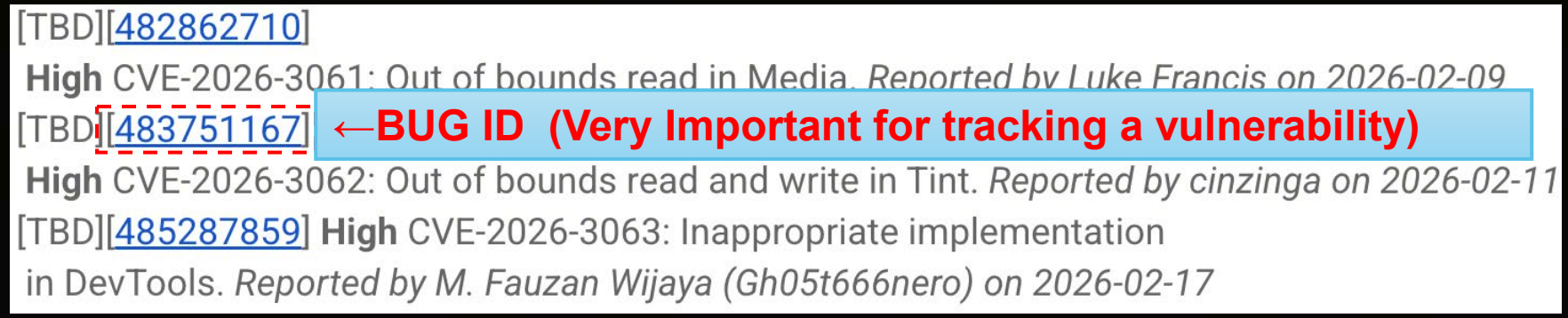
下午3:27 · 2025年7月1日 · **3.2万** 查看 ← **3:27 PM, 1/7/2025, 32k views**

BUILD YOUR OWN SYSTEM IN 5 STEPS

1. Intelligence Analysis
2. Exploit Detection
3. Mitigations Matching
4. Mitigation Decision
5. Protection Enforcement

TRACK NEW BUGS

- Monitor https://chromereleases.googleblog.com/ for new vuln

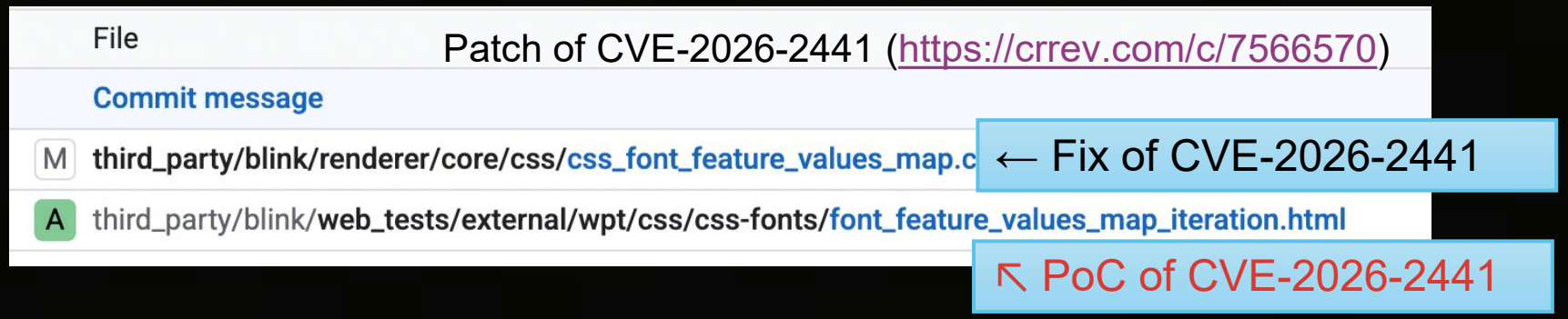


- Visit https://crbug.com/[BUGID] for details
  - Discussions between reporter and developer: parse by LLM

WHAT DOES THE PATCH SAY

- **git log --grep="Bug: BUGID"** to get all related patches

PoC



WHAT DOES THE PATCH SAY

- **git log --grep="Bug: BUGID"** to get all related patches

PoC

Patch of CVE-2026-2441 (https://crrev.com/c/7566570)

	File
	Commit message
M	third_party/blink/renderer/core/css/css_font_feature_values_map.c
A	third_party/blink/web_tests/external/wpt/css/css-fonts/font_feature_values_map_iteration.html

← Fix of CVE-2026-2441

↖ PoC of CVE-2026-2441

Patch

```
 PairSyncIterable<CSSFontFeatureValuesMap>::IterationSource*
CSSFontFeatureValuesMap::CreateIterationSource(ScriptState*) {
-  return MakeGarbageCollected<FontFeatureValuesMapIterationSource>(*this,
-                                                                   aliases_);
+  return MakeGarbageCollected<FontFeatureValuesMapIterationSource>(
+      *this, aliases_ ? *aliases_ : FontFeatureAliases());
 }
```

← Critical Function (Crash Monitoring)

↖ Vulnerable code path

← Explains why vulnerability exists

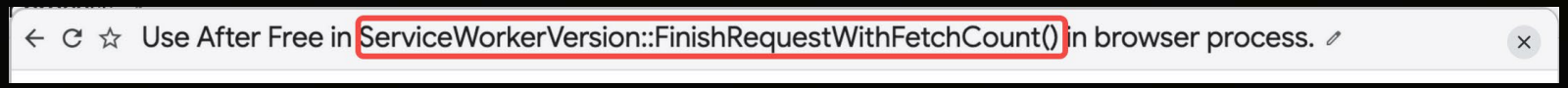
BUILD YOUR OWN SYSTEM IN 5 STEPS

1. Intelligence Analysis
2. Exploit Detection
3. Mitigations Matching
4. Mitigation Decision
5. Protection Enforcement

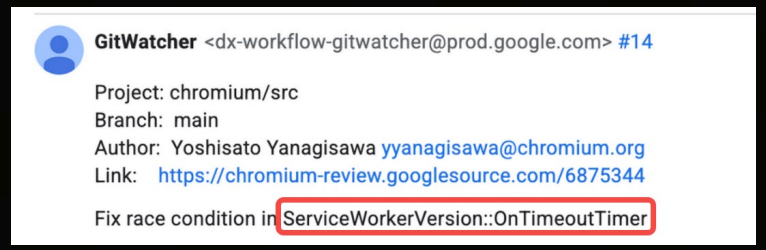
GET THE SIGNATURE OF A VULNERABILITY WITHOUT A POC

Enough data + AI (LLM) = Solution

From Monorail



From Git Log



From Patch

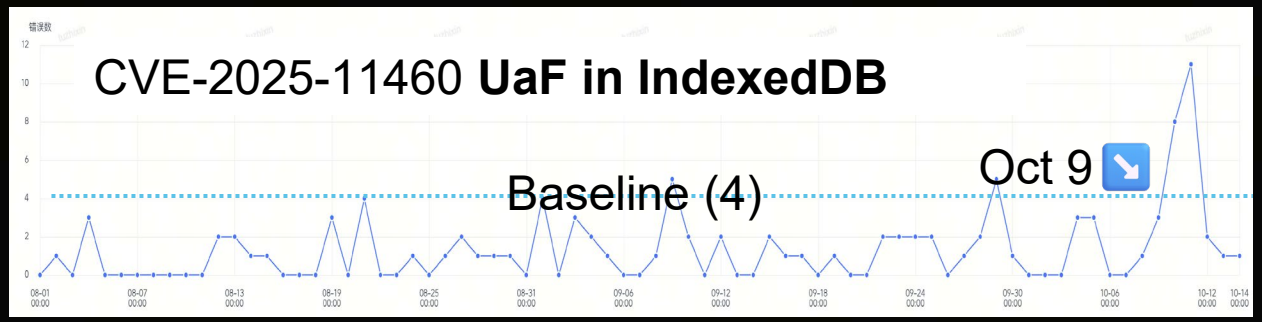
```
 PairSyncIterable<CSSFontFeatureValuesMap>::IterationSource*
CSSFontFeatureValuesMap::CreateIterationSource(ScriptState*) {
-  return MakeGarbageCollected<FontFeatureValuesMapIterationSource>(*this,
-                                                                   aliases_);
```

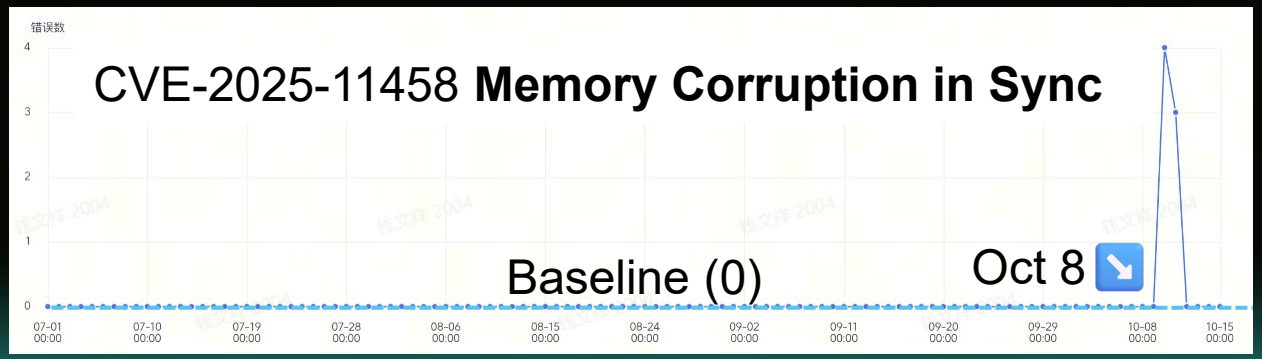
← Monitor this!

← Root cause of the vul

SET-UP MONITOR FOR KEY FUNCTIONS

- Attacker usually run raw PoC to check if vulnerable
- Such ‘failed’ attempts spikes crashes



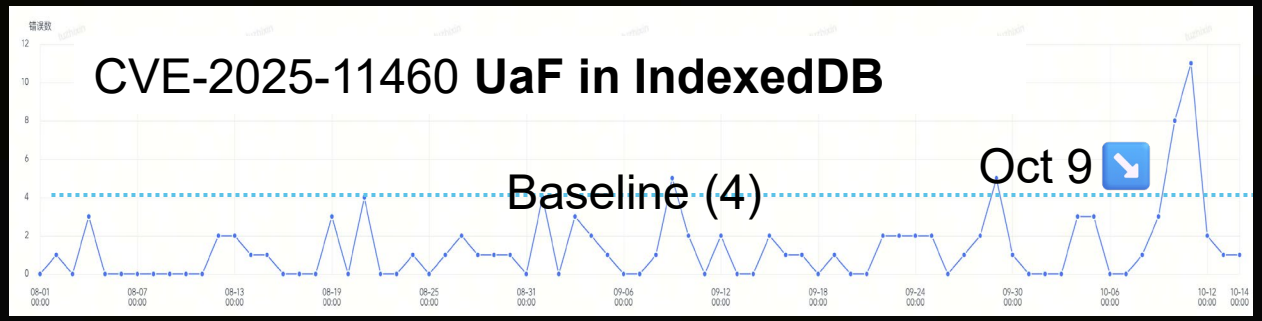


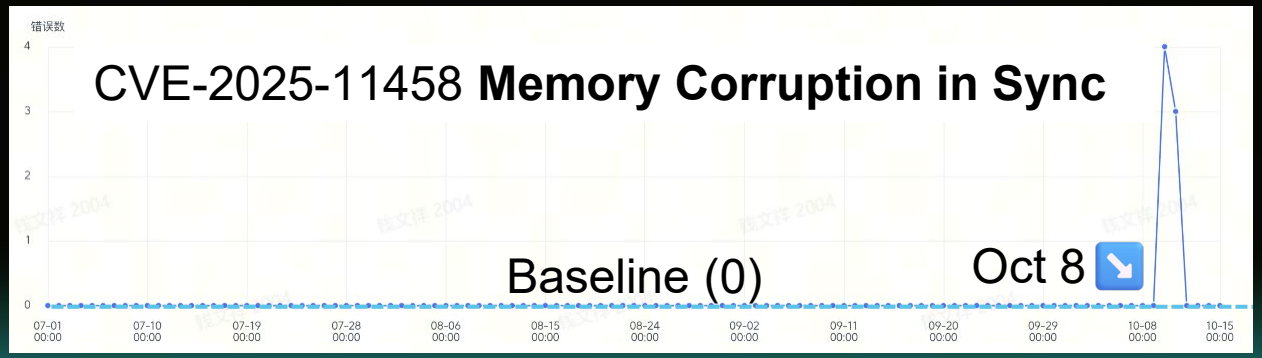
Baseline (2)

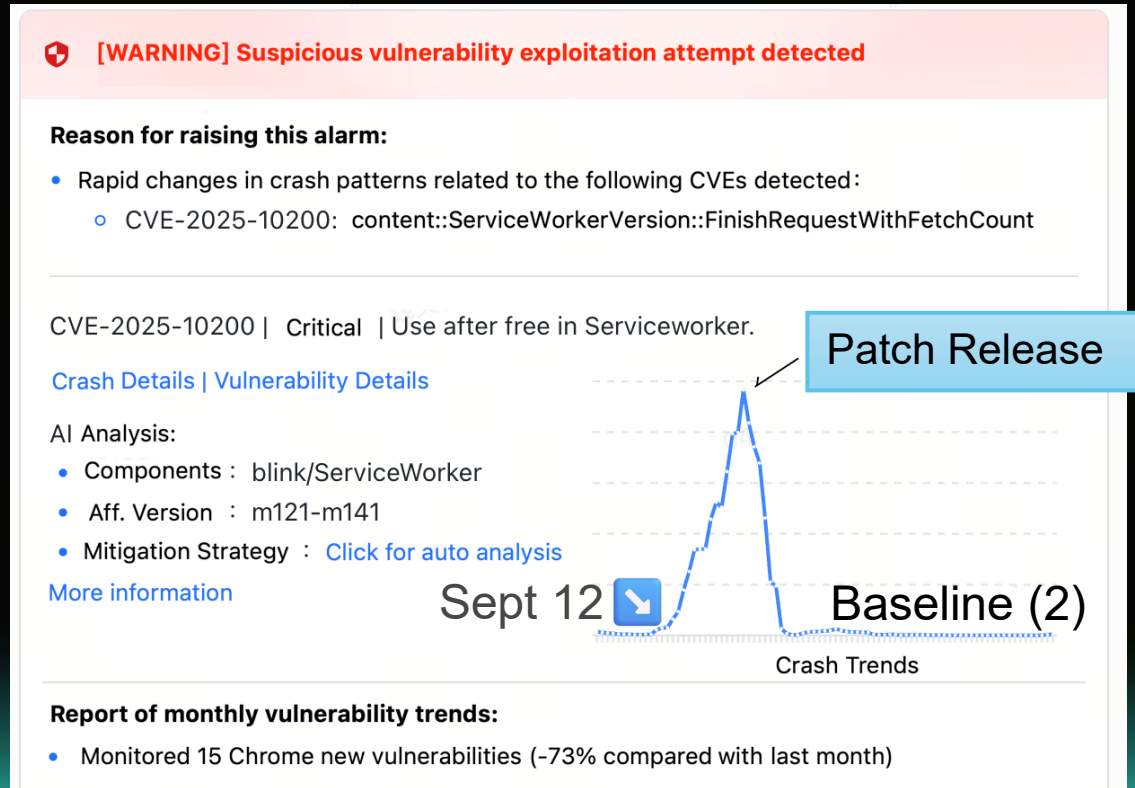
black hat ASIA 2026

SET-UP MONITOR FOR KEY FUNCTIONS

- Attacker usually run raw PoC to check if vulnerable
- Such ‘failed’ attempts spikes crashes
- Easy-to-check/famous N-days = obvious spikes, longer durations







3000 SWITCHES IN CHROMIUM SOURCE

JS Flags
800

Try:
₂, 39ɸĖEŮĵ ˈĖ, Ω̄ ₃, ₂ Ě

```
--untrusted-code-mitigations (Enabl
      type: bool  default: false
--expose-wasm (expose wasm interfac
      type: bool  default: true
--assume-asmjs-origin (force wasm c
      type: bool  default: false
--wasm-disable-structured-cloning (
      type: bool  default: false
--wasm-num-compilation-tasks (numbe
      type: int  default: 10
--wasm-write-protect-code-memory (w
      type: bool  default: false
```

3000 SWITCHES IN CHROMIUM SOURCE

JS Flags	Blink Functions	Base Features	Cmdlines
800	1100	800	200

Try:
node -- v8-options

```
--untrusted-code-mitigations (Enabl
      type: bool  default: false
--expose-wasm (expose wasm interfac
      type: bool  default: true
--assume-asmjs-origin (force wasm c
      type: bool  default: false
--wasm-disable-structured-cloning (
      type: bool  default: false
--wasm-num-compilation-tasks (numbe
      type: int  default: 10
--wasm-write-protect-code-memory (w
      type: bool  default: false
```

See:
runtime_enabled_features.json5

```
{
  name: "LanguageDetectionAPIForWorkers",
  public: true,
},
```

See:
; ⊣ ΊΩaI ÊÑ¼4

```
// How often (in seconds) to check for updates. Should only be used f
// purposes.
const char kCheckForUpdateIntervalSec[] = "check-for-update-interval"
```

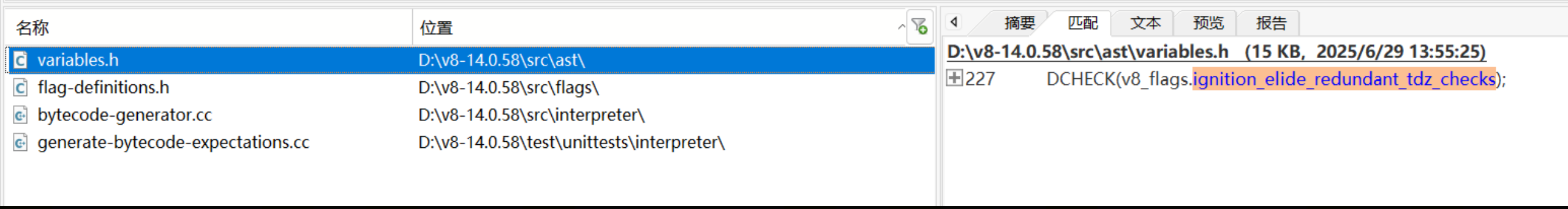
See:
7⁽ Ă ₀ 9ΌĚÜ₃ Ī 7⁽ 9ĚB77

```
49: namespace features {
50: BASE_FEATURE(kUseSmartRefForGPUFenceHandle, base::FEATURE_ENABLED_BY_DEFAULT);
51:

64: // therefore d' == d. QED.
65: BASE_FEATURE(kUseRoundedPointConversion, base::FEATURE_ENABLED_BY_DEFAULT);
66:
67: BASE_FEATURE(kHdrAgtm, base::FEATURE_DISABLED_BY_DEFAULT);
68:
69: BASE_FEATURE(kTransparentIconWorkaround,
70: #if BUILDFLAG(IS_WIN)
```

THE JSFLAG AND IT'S OCCURRENCES

- See `src/flags/flag-definitions.h`
- Parse all JSFlags and locate their usage (v8_flags.*) to determine affected files



WAIT, IS THAT TOO MUCH?

- Lots of POWERFUL switches can eliminate the vulnerability

--jitless
--no-opt
--no-maglev
--no-turbofan
……

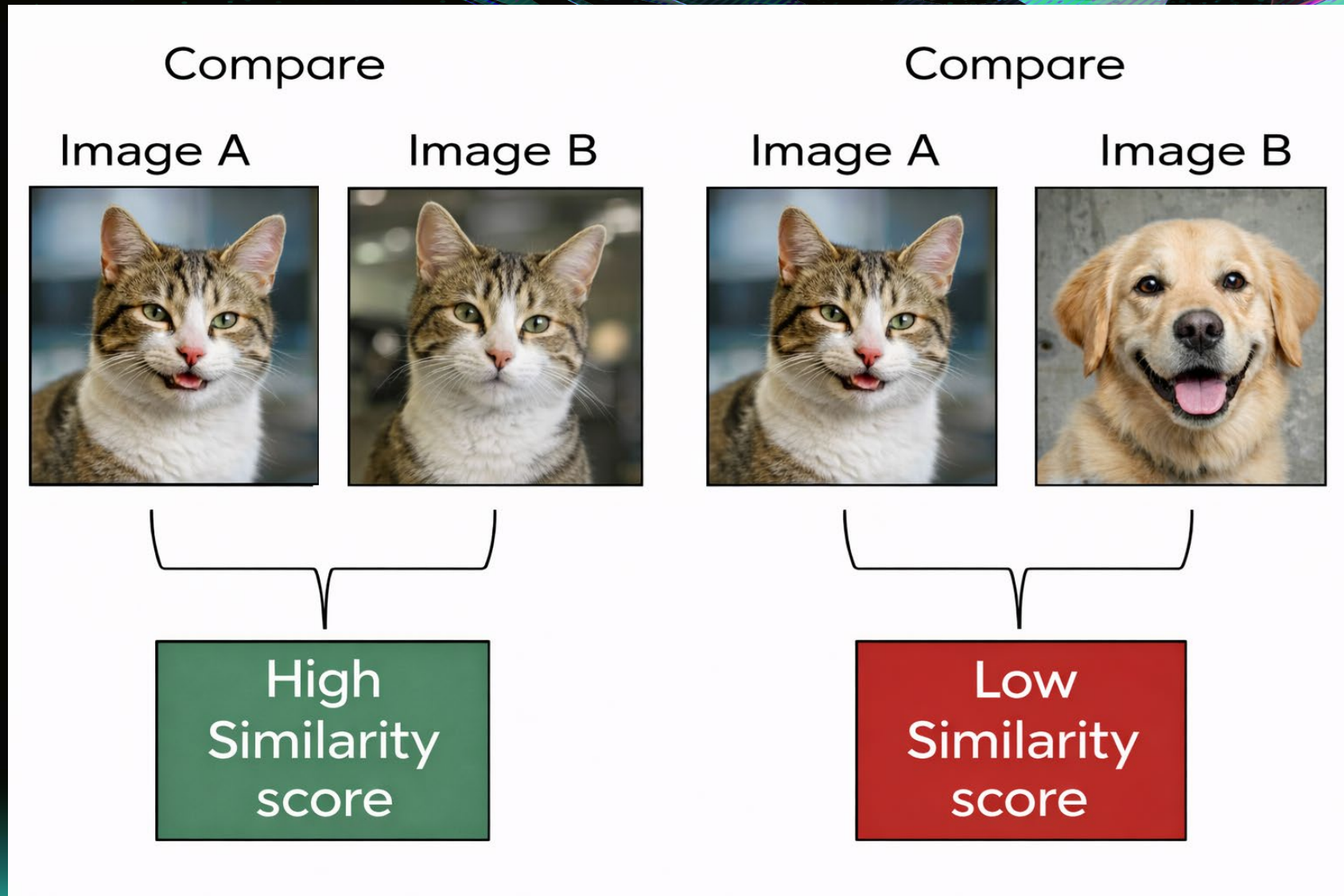
jitless mode not only disable all the JIT of V8 as it shows, it actually disabled ALL THE RWX PAGES. That means WASM IS ALSO DISABLED and may cause many modern website into a blank page due to JS error.

- But some of them are just toooooooo heavy

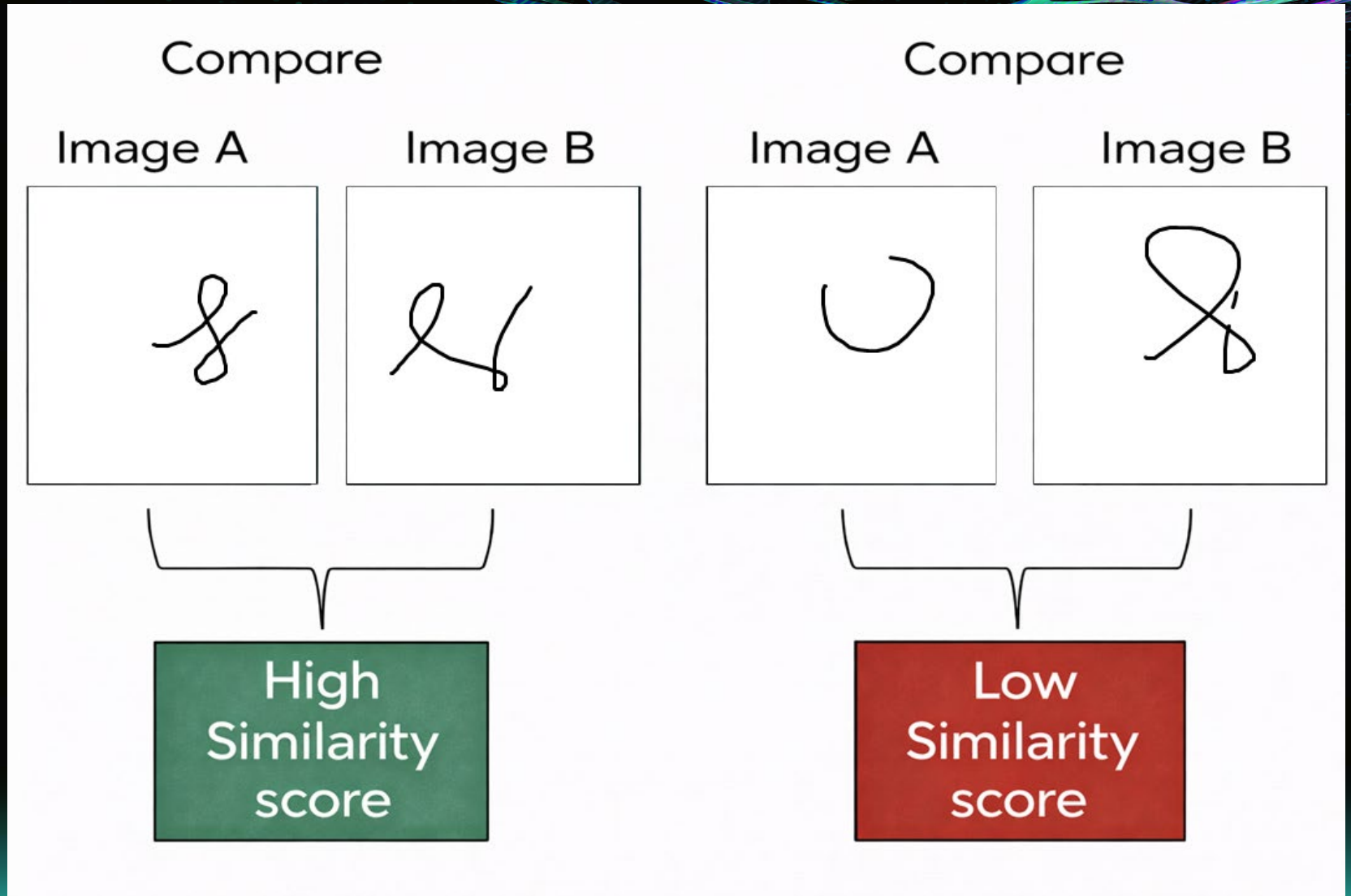
BUILD YOUR OWN SYSTEM IN 5 STEPS

1. Intelligence Analysis
2. Exploit Detection
3. Mitigations Matching
4. Mitigation Decision
5. Protection Enforcement

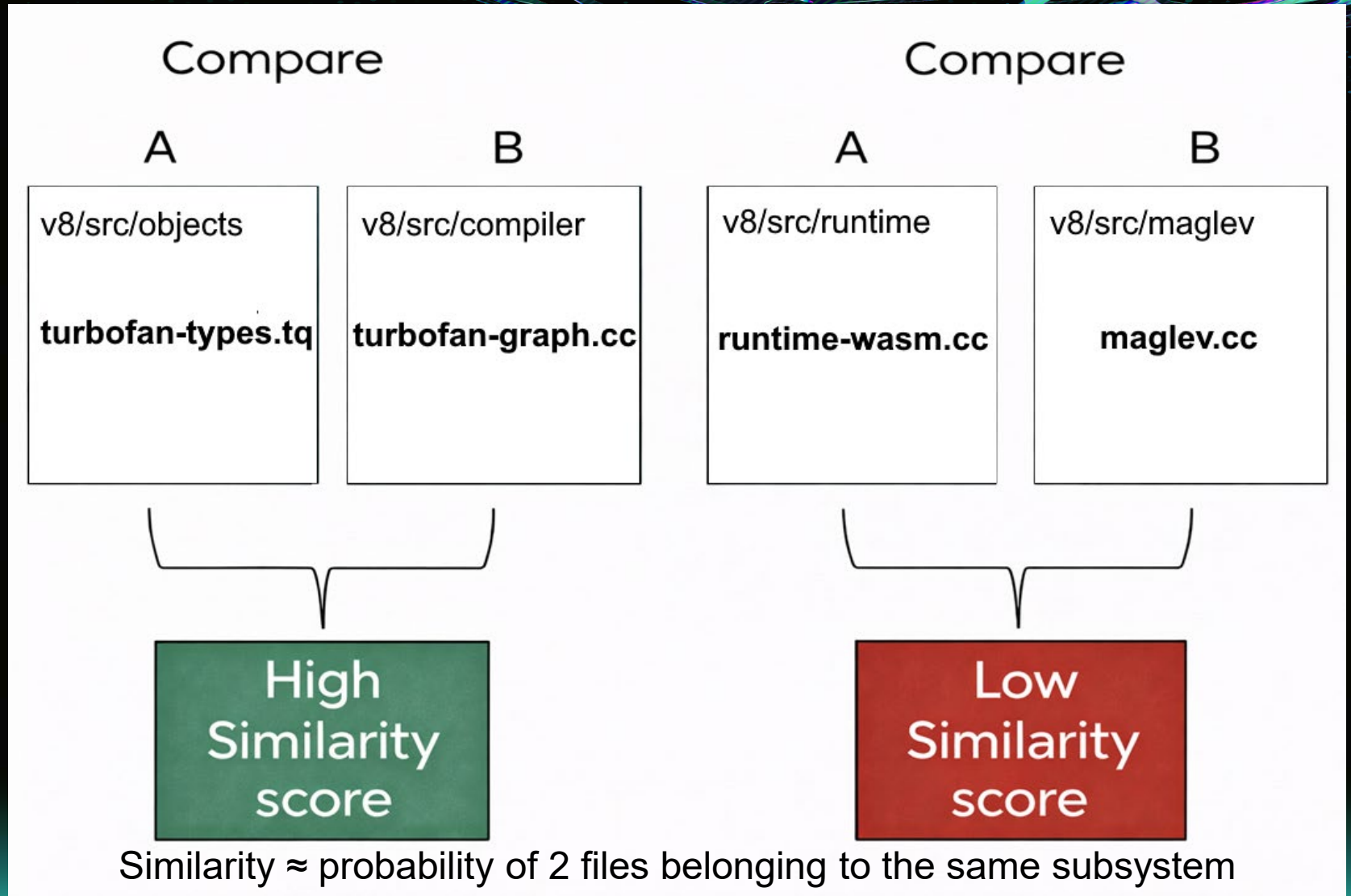
LET'S BEGIN WITH A .. CV PROBLEM



YOU DON'T EVEN NEED TO KNOW THEM TO TELL IF THEY ARE SIMILAR



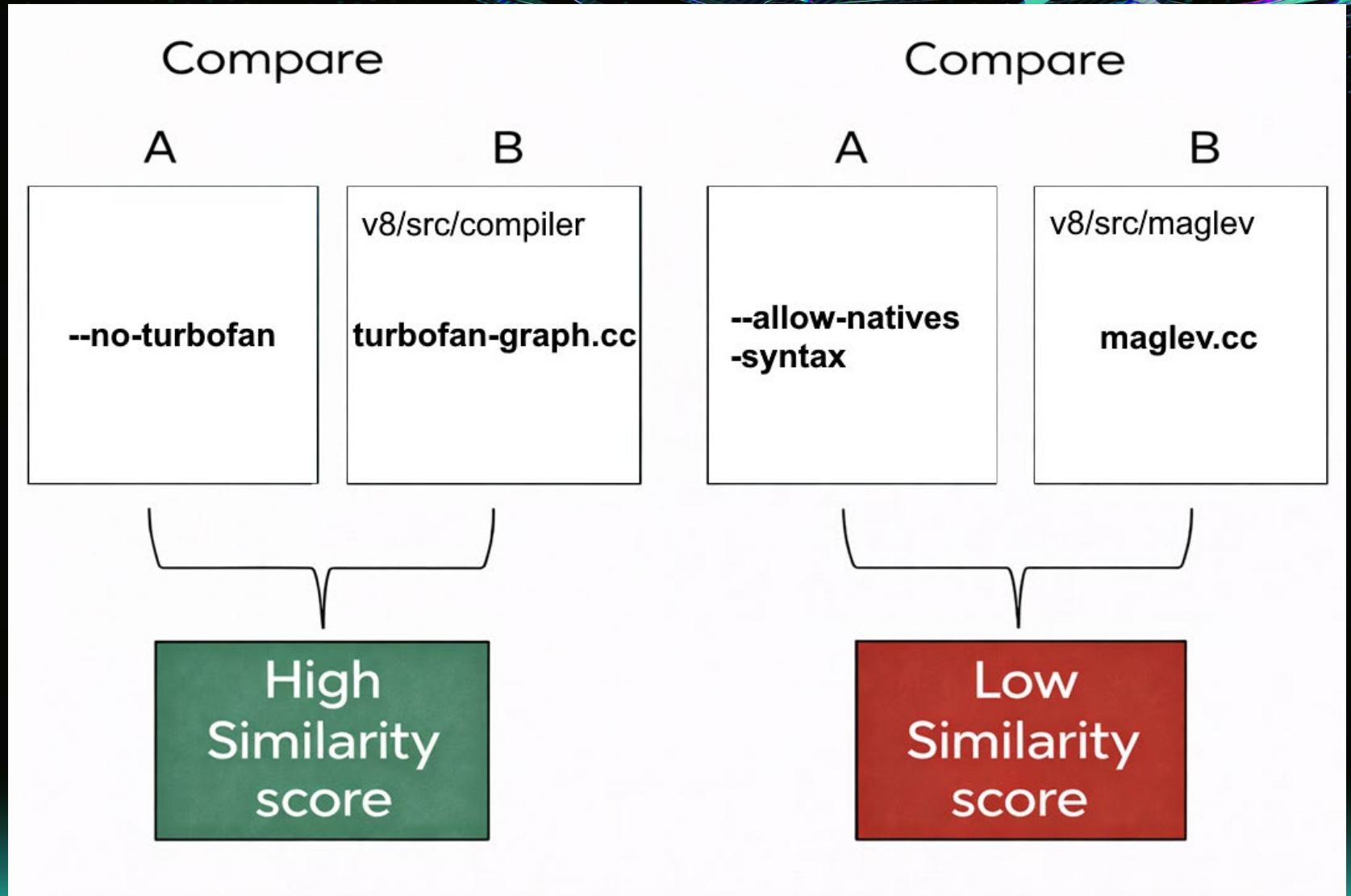
THEY ARE RELATED: SOURCE FILES



Similarity ≈ probability of 2 files belonging to the same subsystem

black hat ASIA 2026

RELATED: A JSFLAG AND THE SOURCE FILE



RELATED: VULNERABILITY AND ITS PATCH

Compare

CVE	PATCH
CVE-2025-5419 [0-day] Arbitrary read/write within the V8 sandbox via Turbofan	v8/src/compiler/turboshaft/ **store-store-elimination-reducer-inl.h**

Files:

- M src/compiler/turboshaft/store-store-elimination-reducer-inl.h

Link: https://chromium-review.googlesource.com/6599190

High Similarity score

Compare

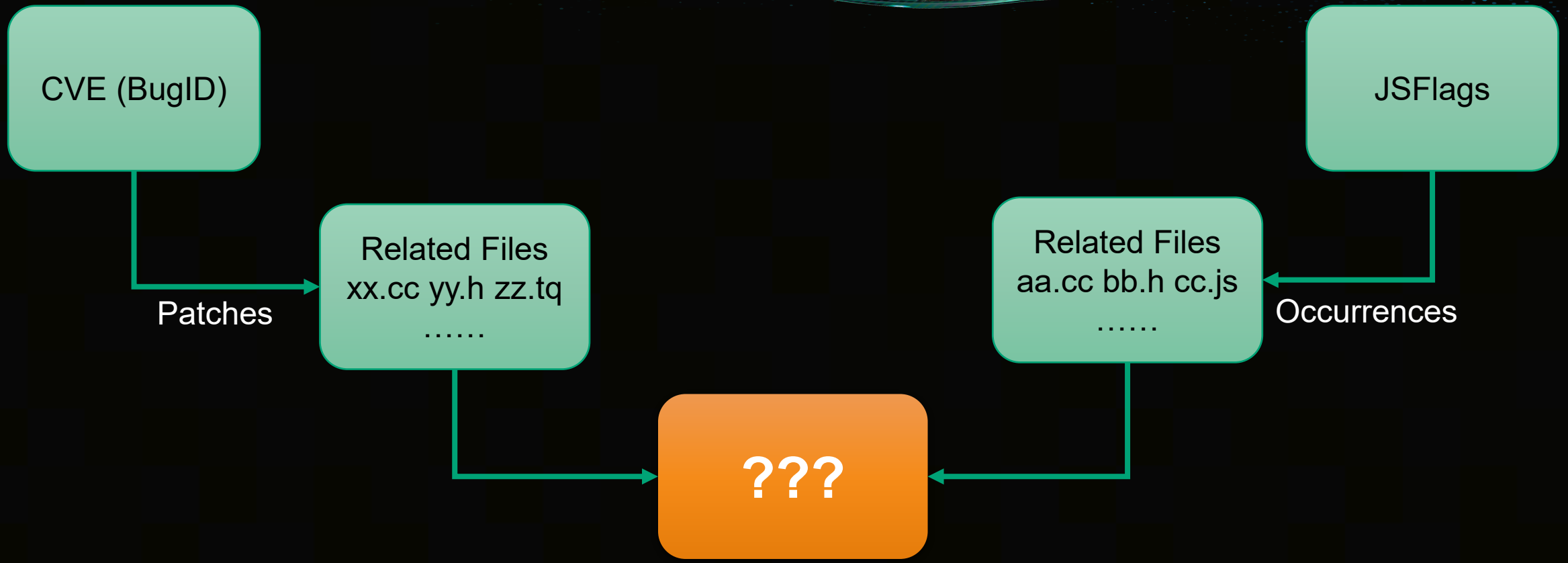
CVE	PATCH
CVE-2024-4947 RCE in V8 maglev	v8/src/maglev **maglev-graph-builder.cc**

M src/compiler/access-info.cc
M src/maglev/maglev-graph-builder.cc

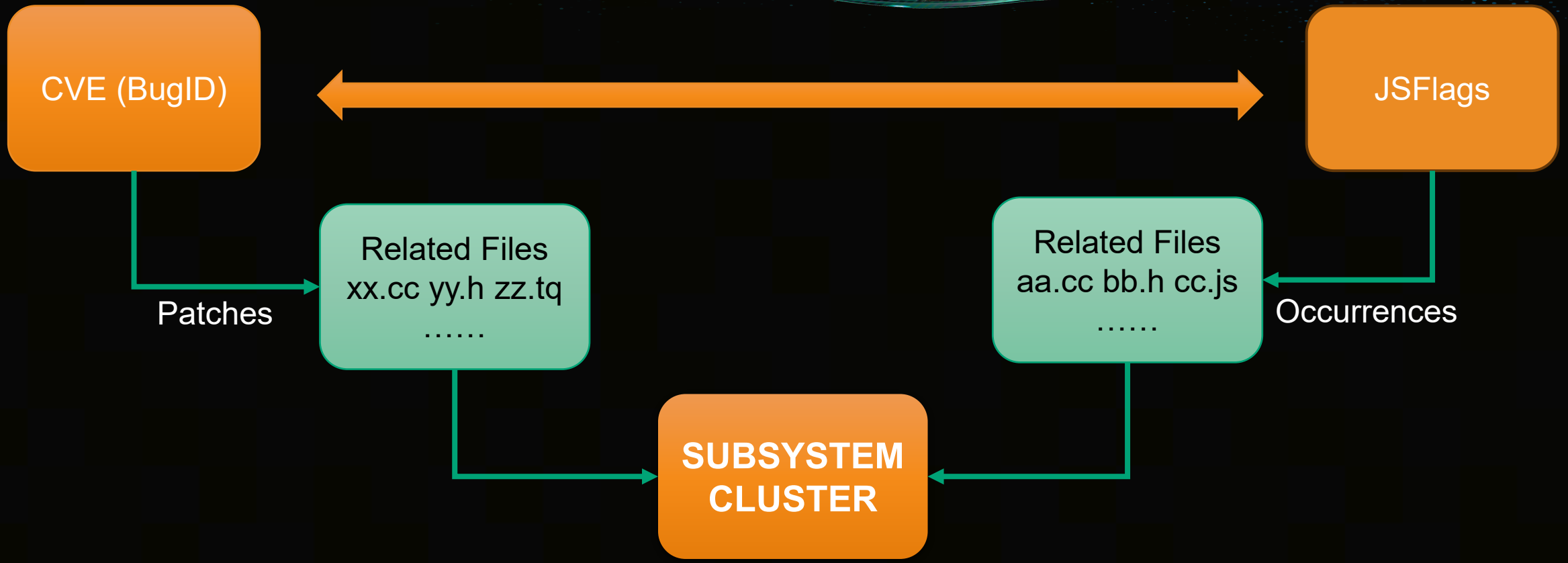
https://chromium-review.googlesource.com/5536090

High Similarity score

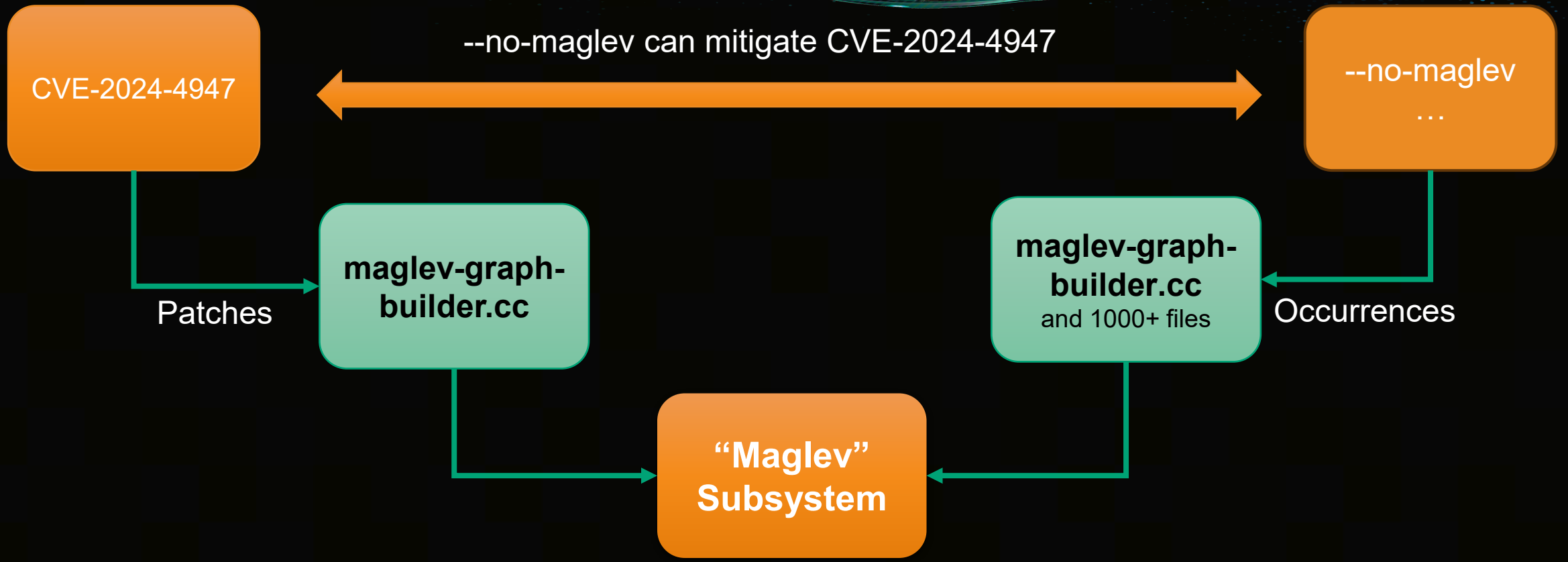
LET'S CHAIN THESE THINGS TOGETHER



BRIDGE THROUGH CVE AND JSFLAGS



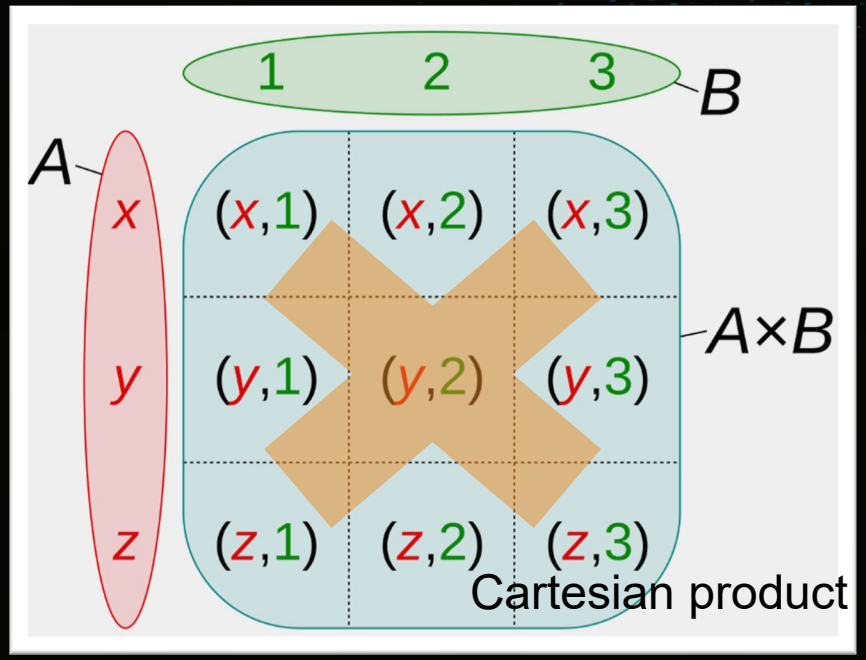
EXAMPLE: CVE-2024-4947



HOW TO CLUSTER A PROJECT, UNSUPERVISED & EFFICIENTLY?

Prob. Scale
> 300 M

- Task: Calculate the relationship between files
- The Cartesian product (pairwise ordered) of v8 Project
  - 18K files × 18K files ≈ 311M ordered pairs
- That's too much!



LET FILES MAKE FRIENDS FIRST

Prob. Scale
> 300 K

- Iterates through all commits and records all co-existed files
- Files in same commit makes them a friend (Related)
- Commit message explains how they become “friends”

id	file_path	commit_hash	commit_msg	co_changed_files
277020	src\snapshot\object-deserializer.cc	8bf15bf1aaca5de...	Reland "[Compiler] Remove code aging support."	src/snapshot/object-deserializer.cc
277021	src\snapshot\object-deserializer.cc	42d3d36bc3b4e7...	Revert "[Compiler] Remove code aging support."	src/snapshot/object-deserializer.cc
277022	src\snapshot\object-deserializer.cc	a205117c6ec6aeb...	[Compiler] Remove code aging support.	src/snapshot/object-deserializer.cc
277023	src\snapshot\object-deserializer.cc	748b717763c205...	[snapshot] Replace deserializer Lists with std::vector.	src/snapshot/object-deserializer.cc
277024	src\snapshot\object-deserializer.cc	95f868595cfb22...	[snapshot] Simplify deserializer APIs	src/snapshot/object-deserializer.cc
277025	src\snapshot\object-deserializer.cc	2e2069cdf61429...	[snapshot] Continue splitting up Deserializer	src/snapshot/object-deserializer.cc
277026	src\snapshot\object-deserializer.cc	b08b63fc82bb26...	[snapshot] Begin splitting up Deserializer	src/snapshot/object-deserializer.cc
277027	src\snapshot\object-deserializer.cc	b086856f0a1537...	[prettyprinter] improve call-printing of GetIterator nodes	test/message/get-iterator1.js
277028	src\snapshot\object-deserializer.cc	59c9e6ff6915f84...	[modules] Fix bug in Module::Instantiate.	test/message/modules-skip-cycle5.js
277029	src\snapshot\object-deserializer.cc	454816f08f8c4f5...	Report unexpected lexical decl also without destructuring	test/message/let-lexical-declaration-destructuring-brace-in-si...
277030	src\snapshot\object-deserializer.cc	94bf354af55261a...	Raise SyntaxError on let [starting an ExpressionStatement	test/message/let-lexical-declaration-in-single-statement.js
277031	src\snapshot\object-deserializer.cc	d33dc16f435c52...	Add test case for Number.prototype.toString (r42364).	test/mjsunit/regress/regress-crbug-679841.js
277032	src\snapshot\object-deserializer.cc	aff64e9dfa436f6...	Parser: Fix InitializerRewriter.	test/mjsunit/regress/regress-679727.js
277033	src\snapshot\object-deserializer.cc	5662f99b998a78...	[turbofan] Teach escape analysis about StringCharAt	test/mjsunit/regress/regress-crbug-677757.js
277034	src\snapshot\object-deserializer.h	6f898234b10a9b...	Reland "[offthread] Template deserializer on Isolate"	src/snapshot/object-deserializer.h
277035	src\snapshot\object-deserializer.h	c73d759bea0359...	Revert "[offthread] Template deserializer on Isolate"	src/snapshot/object-deserializer.h

TF-IDF TO MARK “CLOSE FRIENDS”

Prob. Scale
> 30 K

- Build a global vocabulary first (top 3000 terms from all commit message)
- TF-IDF highlights the most representative terms for each file

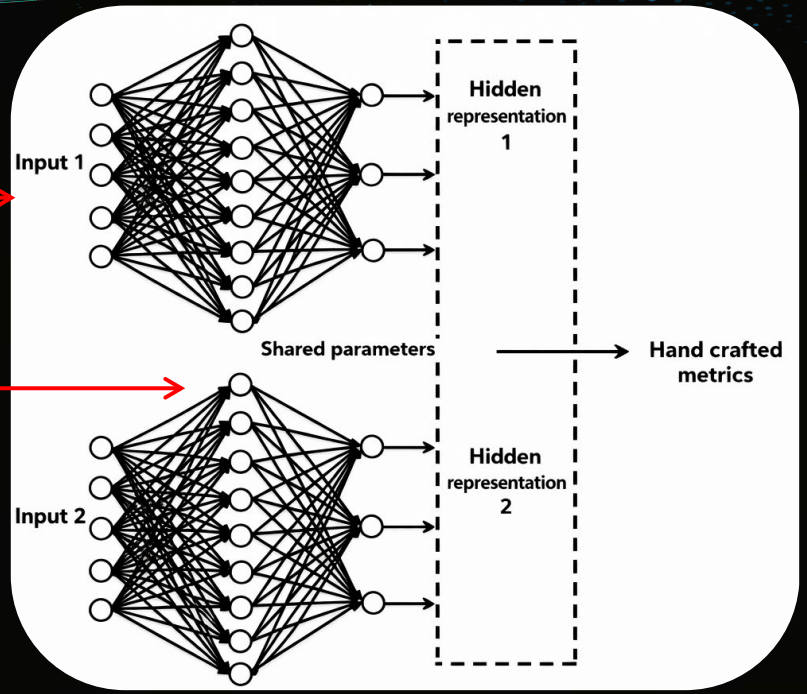
$$\text{TF-IDF} = \text{term frequency} \times \log\left(\frac{\text{total documents}}{\text{documents containing the term}}\right)$$

- Measures how close two files are
  - Files are close if they share similar weighted terms

PULLS CO-CHANGED FILES CLOSER AND UNRELATED FILES APART

Prob. Scale
> 18 K

- Siamese Network are born to do this
  - Two inputs
  - Two identical networks
  - Trained with pairwise TF-IDF samples



PULLS CO-CHANGED FILES CLOSER AND UNRELATED FILES APART

Prob. Scale
> 18 K

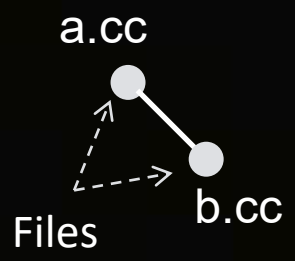
- Input:
  - TF-IDF vector ✕ 2, 3000-dim. each
- Output:
  - 64-dim. representation of each file
  - Distance: L2 distance

$$\text{Euclidean Distance} = |X - Y| = \sqrt{\sum_{i=1}^{i=n} (x_i - y_i)^2}$$

LAST STEP: CLUSTERING

Prob. Scale
> 9 K

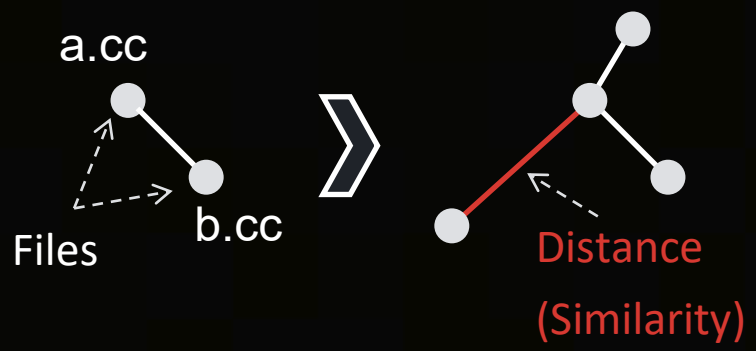
- Starts with any two files
- Distances equals to similarity between each two files



AGGLOMERATIVE CLUSTERING

Prob. Scale
< 9 K

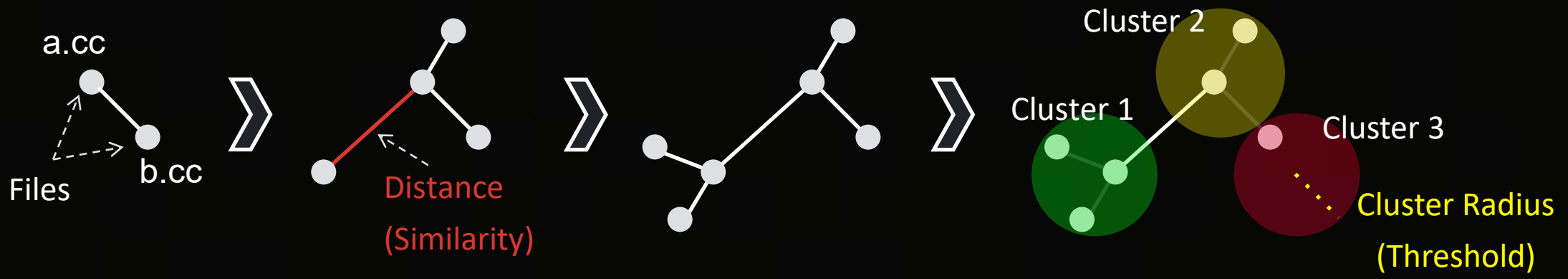
- Keep adding the remaining files into the universe



AGGLOMERATIVE CLUSTERING

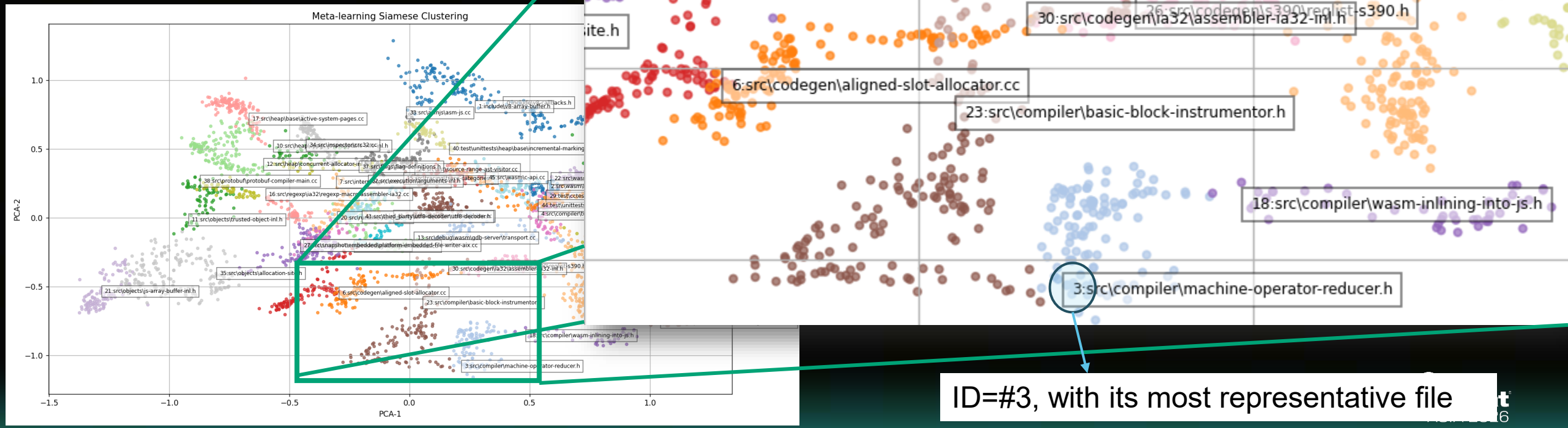
Prob. Scale
> 30

- Until all files are added into the universe
- Define a distance threshold
- Files are clustered



THE UNIVERSE OF SOURCE FILES

- Meta-learning can unsupervised cluster any project sources
- Each subsystem has its own Cluster ID



OPEN-SOURCED AT GITHUB

- We open-sourced the model & tools in Github
- Check it out here:

https://github.com/toolbay/project-cluster

- Trained with CPU, no GPU needed

LET’S TRY IT WITH CVE-2026-4457

- [HIGH] Type Confusion in V8 -- Related with turboshaft
- Released 18 March 2026

repo:v8/v8 488803413

Filter by

- Code 1
- Issues 0
- Pull requests 0
- Discussions 0
- **Commits 2**
- Packages 0
- Wikis 0

Advanced

- Organization
- Author

2 results (257 ms) in v8/v8

v8/v8

[turboshaft] Take use-count saturation into account in IsOnlyUserOf

Fixed: **488803413** Change-Id: I7cf0911de2338144e2d972a53c5787f186b43623 Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/762...

DadaIsCrazy authored and V8 LUCI CQ committed on 3月2日 · 23ec84a

v8/v8

[turboshaft] Make it harder to forget about use-count saturation

We recently had a bug (https://crbug.com/488803413) where we forgot to take saturation into account when looking at a use count. This CL ...

DadaIsCrazy authored and V8 LUCI CQ committed on 3月9日 · 9eec6b0

```
→  defender git:(131.0.6778.268) ✗ PYTHONPATH=. /home/shuni/code/chromium/depot_tools/vpython3 runme.py infer \
  --model-db ./model/jsflags_defender_model.db \
  --patch ../patches/ed25514.diff \
  --top-k 5 \
--showtime
```

00:00:00 / 01:00:00
结束录制

BUILD YOUR OWN SYSTEM IN 5 STEPS

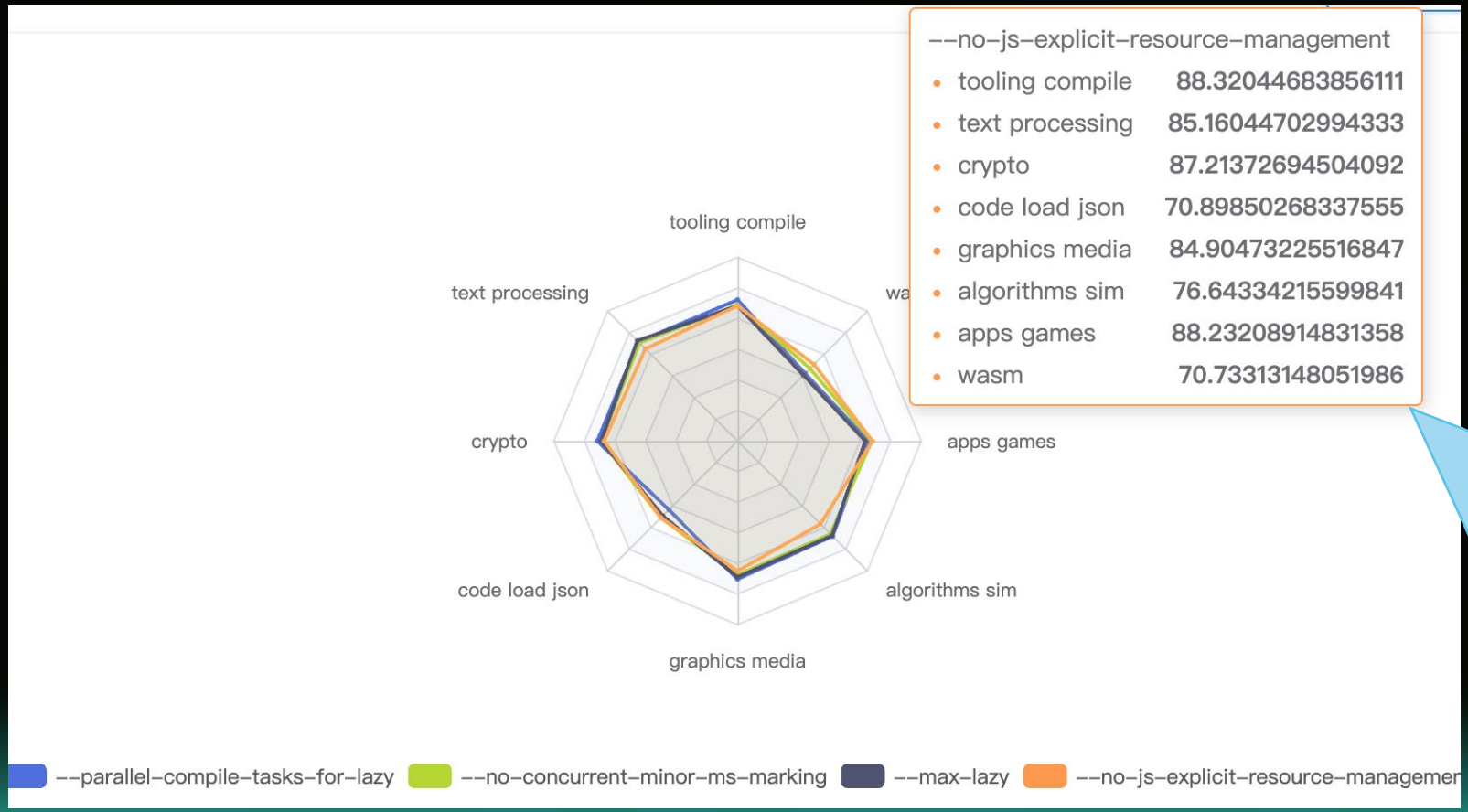
1. Intelligence Analysis
2. Exploit Detection
3. Mitigations Matching
4. Mitigation Decision
5. Protection Enforcement

MANY JSFLAS TO ONE CVE

- We may have lots of JSFlags to one CVE, which one should we use?
- You don’t want interrupt your customer too much
  - Attackers may try to attack you sometime
  - Performances will annoy your customer all the time
- Try to balance them?

GET THE PERFORMANCE DATA

- Pretest all combinations of up to two flags on Jet2Stream
  - $C(800, 2) + C(800, 1) \approx 320{,}400$ combinations total



--no-js-explicit-resource-management

Average **~20%** Performance Regression

Worst **~30%** Regression

black hat ASIA 2026

SORT THE PERFORMANCE DATA

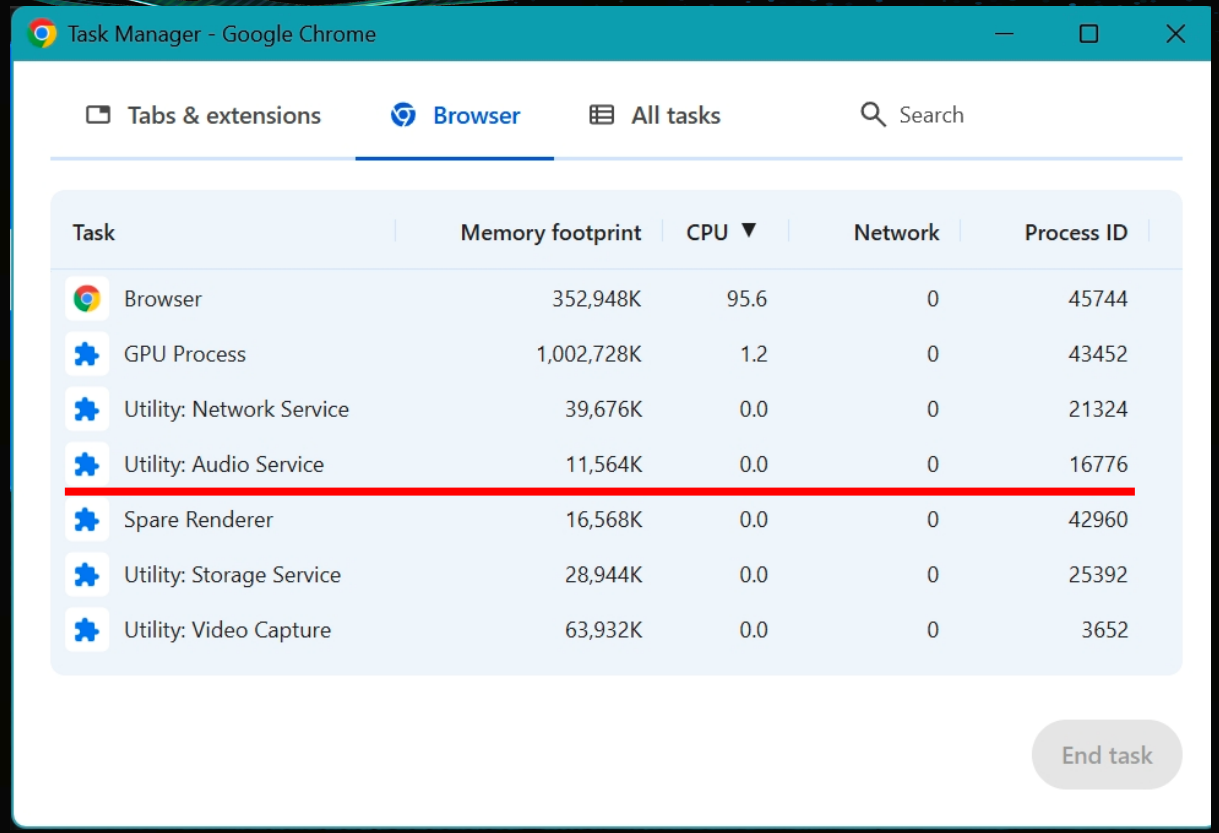
- Ranking all canidates
- Select the one that touches the fewest code paths with the best performance

Overall ranking (F)

Flag	Cluster	Static	Dynamic	Defense points	performance	Overall score
--no-js-explicit-resource-management Pareto	0.100	0.140	-	0.104	92.9 % (0.774)	**0.206**
--no-js-shipping	0.100	-	-	0.077	92.1 % (0.768)	**0.158**
--jitless	0.100	0.400	-	0.154	17.6 % (0.146)	**0.149**

THE SPARE RENDERER PROBLEM

- Spare Renderer
  - pre-warmed renderer process
  - reducing startup latency
  - improving responsiveness
- Conflicts with per-site dynamic JSFlags
- If performance is critical, use command-line flags instead



THE MODIFIED COMMANDLINE

- Append Renderer commandline at startup
- content/browser/renderer_host/render_process_host_impl.cc

```
void RenderProcessHostImpl::AppendRendererCommandLine(
    base::CommandLine* command_line) {
  // Pass the process type first, so it shows first in process listings.
  command_line->AppendSwitchASCII(switches::kProcessType,
                                  switches::kRendererProcess);

  // Call this as early as possible so that --extension-process will show early
  // in process listings. See https://crbug.com/1211558 for details.
  GetContentClient()->browser()->AppendExtraCommandLineSwitches(command_line,
                                                                GetID());
```

BUILD YOUR OWN SYSTEM IN 5 STEPS

1. Intelligence Analysis
2. Exploit Detection
3. Mitigations Matching
4. Mitigation Decision
5. Protection Enforcement

CLIENT-SIDE IMPLEMENTATION BREAKDOWN

To make it work:

- Step 1: Decide per-site flags
- Step 2: Keep process correctness
- Step 3: Inject flags at renderer startup

STEP 1: DECIDE PER-SITE FLAGS

- Get the URL being navigated
- Test if it matches the strategy

```
void ChromeContentBrowserClient::SiteInstanceGotProcessAndSite(
    SiteInstance* site_instance) {
  std::string site_js_flags = jsflags::GetFlagsForSite(
      site_instance->GetSiteURL(),
      site_instance->GetBrowserContext());
  if (site_js_flags.empty()) {
    security_jsflags_by_process_id_.erase(process->GetID());
  } else {
    security_jsflags_by_process_id_[process->GetID()] = site_js_flags;
  }
```

STEP 2: COMPATIABLE WITH THE ORIGINAL JITLESS SUPPORT

- In IsJitDisabledForSite()

```
  if (map && site_url.is_empty()) {
    jit_disabled_by_content_settings =
        map->GetDefaultContentSetting(ContentSettingsType::JAVASCRIPT_JIT,
                                      nullptr) == CONTENT_SETTING_BLOCK;
    return jit_disabled_by_content_settings;
  }
```

Read Original Settings

```
  // Only disable JIT for web schemes.
  if (site_url.SchemeIsHTTPOrHTTPS()) {
    jit_disabled_by_content_settings =
        map && map->GetContentSetting(site_url, site_url,
                                      ContentSettingsType::JAVASCRIPT_JIT) ==
                   CONTENT_SETTING_BLOCK;
  }

  return jit_disabled_by_content_settings ||
         jsflags::IsJitlessForSite(site_url, browser_context);
}
```

Merge JSFlags with user provided --jitless

STEP 2: Keep process correctness

- In ShouldUseSpareRenderProcessHost()
- Disable spare processes for matched navigation

```
// Spare renderers launch before site-specific renderer switches can be
// computed. Force a fresh renderer when this site has dynamic JS flags.
if (!jsflags::GetFlagsForSite(site_url, browser_context).empty()) {
  return SpareProcessRefusedByEmbedderReason::V8OptimizationsDisabled;
}
```

PROBLEM: WHERE TO MODIFY V8 Flags?

- You can’t modify JS flags after V8 is inited
- V8 Initialize() function reads global command line flags and set-up V8

```
void V8Initializer::Initialize(IsolateHolder::ScriptMode mode,
                               const std::string& js_command_line_flags,
                               v8::OOMErrorCallback oom_error_callback) {
  static bool v8_is_initialized = false;
  if (v8_is_initialized)
    return;

  // Flags need to be set before InitializePlatform as they are used for
  // system instrumentation initialization.
  // See https://crbug.com/v8/11043
  SetFlags(mode, js_command_line_flags);
```

STEP 3 INJECT FLAGS AT RENDERER STARTUP

- Use child process ID as key to access customized flags
- Merge them at renderer startup so V8 can use them

chrome > browser > chrome_content_browser_client.cc > ...

```
2751  void ChromeContentBrowserClient::AppendExtraCommandLineSwitches(
```

```
2828        auto jsflags_it = security_jsflags_by_process_id_.find(child_process_id);
2829        if (jsflags_it != security_jsflags_by_process_id_.end()) {
2830          std::string merged_js_flags = MergeJavaScriptFlags(
2831              command_line->GetSwitchValueASCII(blink::switches::kJavaScriptFlags),
2832              jsflags_it->second);
2833          if (!merged_js_flags.empty()) {
2834            command_line->AppendSwitchASCII(blink::switches::kJavaScriptFlags,
2835                                            merged_js_flags);
2836          }
2837        }
2838      }
2839      security_jsflags_by_process_id_.erase(child_process_id);
```

RENDERER STARTED WITH DIFFERERNT JSFLAGS

- Each process is launched with its own JSFlags
- Protections take effect immediately once the client receives the strategy

```
/home/shuni/code/chromium/src/out/ReleaseLite/chrome --type=renderer
omium-dev-profile --js-flags=--noopt --change-stack-guard-on-fork=ena
```

```
/home/shuni/code/chromium/src/out/ReleaseLite/chrome --type=renderer
omium-dev-profile --js-flags=--no-maglev --change-stack-guard-on-for
```

```
/home/shuni/code/chromium/src/out/ReleaseLite/chrome --type=renderer
omium-dev-profile --js-flags=--no-turbofan --change-stack-guard-on-fo
```

TEMOPRARY JSFLAGS ARE NOT THE FINAL FIX

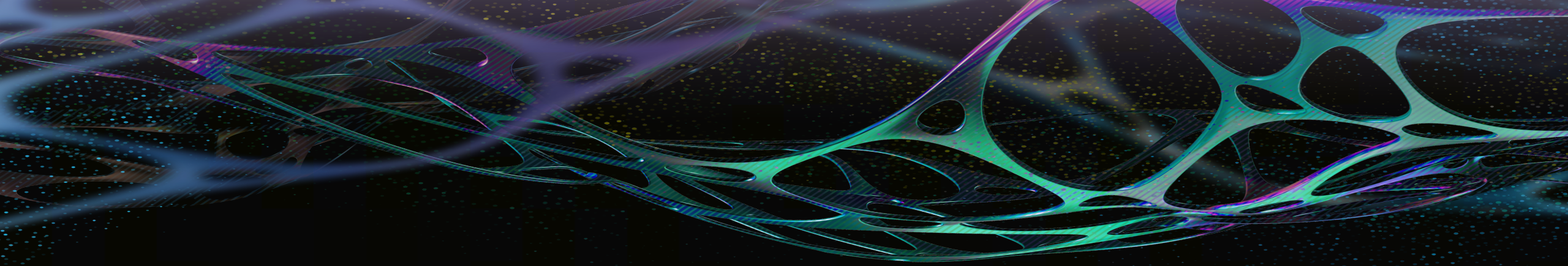
- Continue applying upstream N-day patches
- **git log --grep="Bug: BUGID"** and apply them to your project
- Choose patches from the closest version to your build to minimize conflicts

Cherry picks

refs/branch-heads/7204: [M138-LTS] safe_browsing: Fix UAF in RendererURLLoaderThrottle	(Merged)
refs/branch-heads/7390: [M141] safe_browsing: Fix UAF in RendererURLLoaderThrottle	(Merged)
refs/branch-heads/7444: [M142] safe_browsing: Fix UAF in RendererURLLoaderThrottle	(Merged)

TAKEAWAYS

- Build automated pipelines to keep pace with AI-driven vulnerability discovery and exploitation
- Mitigation is temporary, don’t stop patching
- Keep client and Chromium versions up to date
- Enable Chromium’s built-in protections (sandbox, etc.)
- Download the tool here:
  - https://github.com/toolbay/project-cluster



Thank you for listening!