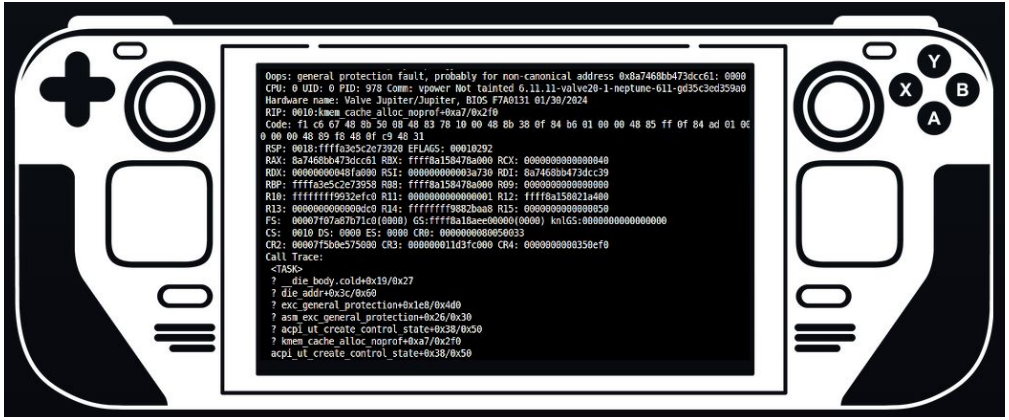
Proactive and crash-time data collection on Steam Deck



```
Oops: general protection fault, probably for non-canonical address 0x8a7468bb473dcc61: 0000
CPU: 0 UID: 0 PID: 978 Comm: vpower Not tainted 6.11.11-valve20-1-neptune-611-gd35c3ed359a0
Hardware name: Valve Jupiter/Jupiter, BIOS F7A0131 01/30/2024
RIP: 0010:kmem_cache_alloc_noprof+0xa7/0x2f0
Code: f1 c6 67 48 8b 50 08 48 83 78 10 00 48 8b 38 0f 84 b6 01 00 00 48 85 ff 0f 84 ad 01 00
0 00 00 48 89 f8 48 0f c9 48 31
RSP: 0018:ffffa3e5c2e73920 EFLAGS: 00010292
RAX: 8a7468bb473dcc61 RBX: ffff8a158478a000 RCX: 0000000000000040
RDX: 0000000048fa000 RSI: 000000000003a730 RDI: 8a7468bb473dcc39
RBP: ffffa3e5c2e73958 R08: ffff8a158478a000 R09: 0000000000000000
R10: ffffffff9932efc0 R11: 0000000000000001 R12: ffff8a158021a400
R13: 0000000000000dc0 R14: ffffffff9882baa8 R15: 0000000000000050
FS:  00007f07a87b71c0(0000) GS:ffff8a18aee00000(0000) knlGS:0000000000000000
CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
CR2: 00007f5b0e575000 CR3: 000000011d3fc000 CR4: 0000000000350ef0
Call Trace:
 <TASK>
 ? __die_body.cold+0x19/0x27
 ? die_addr+0x3c/0x60
 ? exc_general_protection+0x1e8/0x4d0
 ? asm_exc_general_protection+0x26/0x30
 ? acpi_ut_create_control_state+0x38/0x50
 ? kmem_cache_alloc_noprof+0xa7/0x2f0
 acpi_ut_create_control_state+0x38/0x50
```

Guilherme G. Piccoli

gpiccoli (at) igalia.com / gpiccoli (IRC/Matrix)

Linux Plumbers Conference 2025 - Tokyo

whoami

- Living in Brazil, applied mathematician that moved to computers!
  - IBM/LTC - HW/SW interactions on PPC64, I/O, etc
  - Canonical/Sustaining Eng. - kernel/ low-level userspace bugs

- Igalia since 2021 - **FOSS cooperative**, upstream focus
  - Outstanding place, ethics and QoL are priorities here!
  - kdump/pstore, HW enablement, bug fixing, etc
  - Started working with Steam Deck to collect kernel crash info

Talk summary

- **Steam Deck intro**
- **Principles of data collection and submission in a gaming console**
- **Crash-time data**: pstore ~~vs~~ && kdump - what can we improve?
- **System events**: OOMs, split locks, GPU issues *et al.*
- **Aggregate and submit time**: Telemetry FTW

Steam Deck, a Linux game console

- **AMD APU (8 cores)**
  - 16G RAM
  - up to 1TB nvme storage
  - (O)LED screen option(s)

- **SteamOS 3**: Arch-based, rolling (immutable) updates, A/B model
- **Sophisticated stack for games**: Proton (Wine + DXVK + VKD3D, etc)
- **Igalia working in many fronts**: userspace graphics, kernel, FexEMU, SteamOS distro for the Deck and also for Steam Frame

Ideals of data collected in the Deck

Two different classes of data: breakages and regular information

- *Engineers* fixing bugs would like to see when their SW/HW breaks
- *Developers* implementing optimizations / features want metrics

Two "realms" of data: system info (kernel/FW) vs userland (games, apps)

- Infrastructure/system data is the focus here
- Though game issues might also be unveiled (indirectly) from system data

Breakages: kernel oops / crashes, GPU hangs, application coredumps

Informational metrics: split locks, OOMs, GPU/mesa traces, pluggable HW

...and how to visualize all the data

- ***`steamos-log-submitter`***: accumulate and send the data to Valve servers
    - systemd service with hooks, quite modular
    - crashes, metrics, GPU, system journal, tracing

- **Telemetry**: observe the data graphically -> Sentry
    - Internal instance for Valve / some contractors
    - Alarms, summary of failures, statistics

All of that, opt-in ✓

If the users wish, they enable the collection mechanisms

Bugs first, metrics later: kernel crashes

- Very likely the **most severe type of failure**: kernel dies
  - null ptrs, HW issues, etc

- SteamOS is configured to crash on oops and soft/hard-lockups
  - *`kdumpst`* tooling (AUR) + *`steamos-kdumpst-layer`* (SKL)
  - pstore **+** kdump

Let’s dive a bit in the technologies

The good ol’ kdump

- **Kexec-based solution**, new kernel collects data from the broken one
  - On panic, jump to a “fresh” preloaded kernel, that collects vmcore

- **Pro:** huge amount of info, vmcore is a snapshot of full kernel memory

- **Cons:**
  - *Pre-reserved memory required* (`crashkernel=`); >200M lately
    - Reserved on boot, can’t adjust without reboot
  - *Size of vmcore* (could be in the GB order)
  - *Privacy:* bunch of kernel data, for good and bad - Deck’s user data!
  - *Risks:* crash kernel booting and data collection
    - PCI devices misbehave, OOMs, makedumpfile bugs

Pstore: a lightweight way

- **Saves the kernel log in a persistent storage** (backend)
  - On Deck: stored on RAM memory
  - Common in embedded devices and Chromebooks

- **Pros:** fast and (hopefully) transparent process
  - “Doesn’t require” memory reservation
  - Multiple front-ends; ChromeOS collects console, for example
  - Bonus points: no kexec support is required!

- **Cons:** way less information collected, only logs
  - Crash logs collected **after** some panic notifiers (risks!)
  - FW might corrupt RAM on boot (not Deck’s case!)

SteamOS approach - why not both?

- **Kdumpst:** defaults to pstore, but also supports kdump
  - In case pstore fails, or the user wants a full vmcore
  - For Arch, it’s a bit tricky (we don’t control the FW)
  - Also, “funny” story: the tool supports both initcpio and dracut **\o/**

- Layer to make it compatible with SteamOS log collecting: SKL
  - *steamos-kdumpst-layer*
  - Change some defaults and move logs basically
  - Extra info on panic time (memory and task state, CPUs backtraces)
  - *panic_print* and *crash_kexec_post_notifiers*

What can we improve?

- First of all, panic notifiers can be a problem
  - These lists of callbacks run at certain times, on panic for example
  - Long-time proposed refactor, to improve reliability
  - Ideas from KR presentation related to abandon notifiers infrastructure
- Ramoops: tricky to reserve memory and risky (FW corruption)
  - Well ... **was** tricky, Steven Rostedt improved that with `reserve_mem`
- Graphical panic screen - long-term issue
  - Greatly improved by Jocelyn Falempe, drm panic handler
  - Still not implemented on Deck, alternative idea - UEFI notifier

Back to bugs - GPU resets

Another type of interesting / relevant failure

- The amdgpu driver provides **some** level of info in case the GPU hangs
- Complex topic - long-term work from AMD and Igalia
    - Our colleagues Andre Almeida and Rodrigo Siqueira mostly

Limited scope, hard to correlate with user drivers (mesa)

- Useful for determine issues on GFX layers (Proton as in DXVK, e.g.)

Other metrics - tracing

What about metrics to determine the overall system “health” / optimize?

- Two interesting metrics to collect: OOMs, split locks
- No kernel event-sender for OOMs, parsing logs not ideal…
- Same for **split locks** (atomic access on “split” cache lines)
    - Intel supported first, AMD support is recent.
    - Bus lock - extreme drop in perf. Some games do it, for example
    - Funny story: **https://lwn.net/Articles/911219/**

Idea - use tracing infrastructure

- With that, we can inspect relevant functions / events
  - OOM has a tracing event for *`mark_victim`*
  - Split lock can be instrumented on function level (ftrace)
  - Implemented using tracing instances and *`trace_pipe`* blocked read
  - Other events available, big list on tracefs!

Overall system status - peripherals!

- **Generic snapshot of the system through systemd journal**
  - Often doesn’t include devices’ detailed information

- ***`steamos-systemreport`*** **to the rescue**
  - Collects data like *`lspci`*, *`lsusb`*, audio and network information
  - Battery info, display/monitors details
  - Also collects the systemd journal

Overall system status - peripherals!

- **Coredumps and minidumps are somehow collected too!**
  - Minidumps are application memory snapshots, Steam-driven
  - Not much details on these collections…

- **GPU traces** (and gamescope compositor data as well)
  - Tooling to collect data from tracefs

Now to data submission / observation

- **All of that data mentioned so far comes from different tooling**
  - And different formats - textual, memory dumps

- ***`steamos-log-submitter`* (SLS) is the aggregator / submitter**
  - Work from Vicki Pfau
  - Hooks for the many tools providing the data
  - API to communicate with Sentry, the telemetry service
  - Summarizes data (like the splat on kernel oops)

- **Sentry is then the resource used to visualize all of that**
  - Smart search, categorization, charts, statistics, etc

igalia

Join us!

https://www.igalia.com/jobs

Image credits:
slide #1: https://www.deviantart.com/chanxgg/art/Steam-Deck-Transparent-Design-89408555
slide #4: Liam Dawe/GamingOnLinux, PNG version by VulcanSphere, CC BY-SA 4.0