TU Graz
SCIENCE
PASSION
TECHNOLOGY

Eviction Notice:
Reviving and Advancing Page Cache Attacks

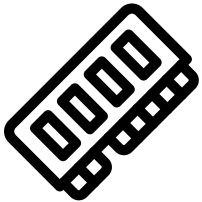
Sudheendra Raghav Neela Jonas Juffinger Lukas Maar Daniel Gruss

Graz University of Technology, Graz, Austria

NDSS 2026

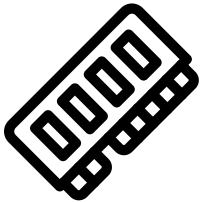
› isec.tugraz.at

The Linux Page Cache



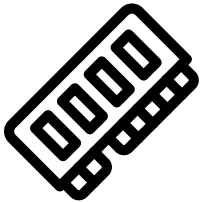
- Operating Systems: Cache pages in memory (RAM)

The Linux Page Cache



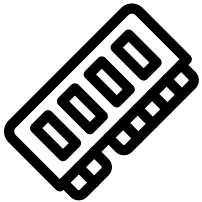
- Operating Systems: Cache pages in memory (RAM)
- Faster Accesses

The Linux Page Cache



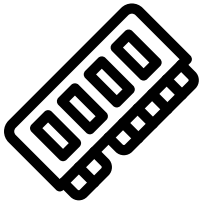
- Operating Systems: Cache pages in memory (RAM)
- Faster Accesses: Shared Libraries

The Linux Page Cache



- Operating Systems: Cache pages in memory (RAM)
- Faster Accesses: Shared Libraries, Binaries

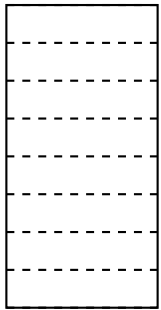
The Linux Page Cache



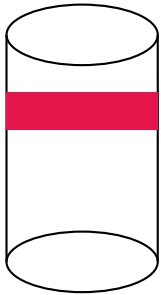
- Operating Systems: Cache pages in memory (RAM)
- Faster Accesses: Shared Libraries, Binaries, Files

The Page Cache

CPU

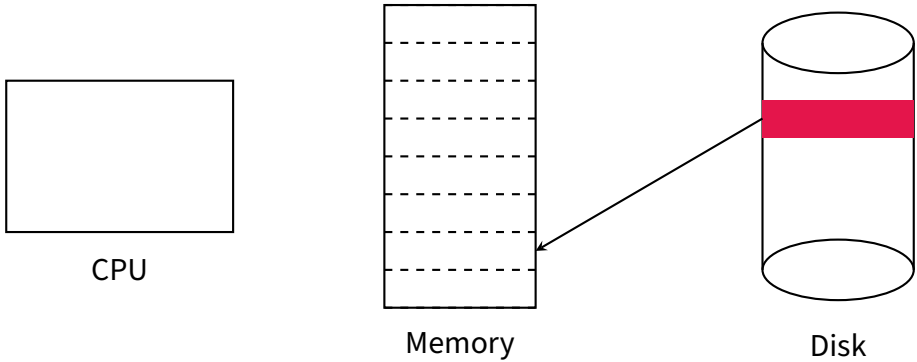


Memory

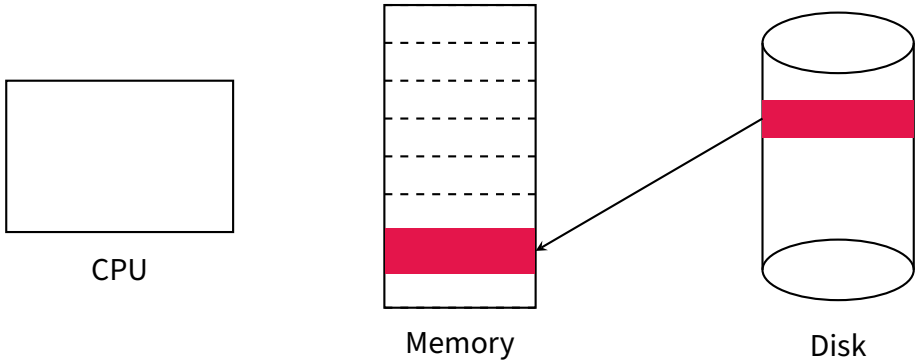


Disk

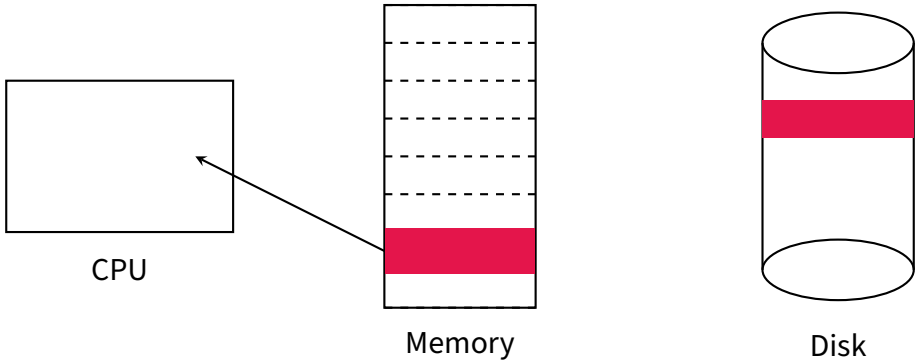
The Page Cache



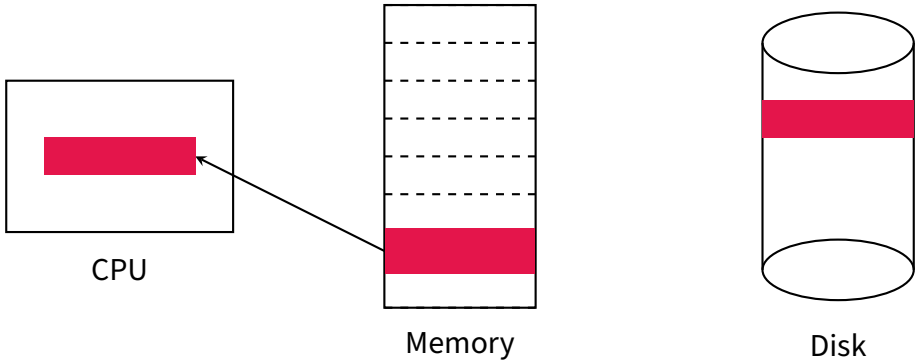
The Page Cache



The Page Cache

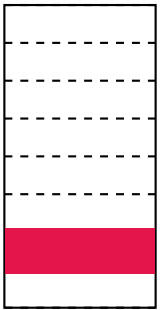


The Page Cache

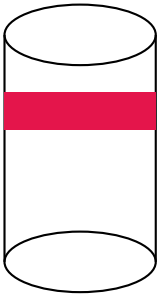


The Page Cache

CPU

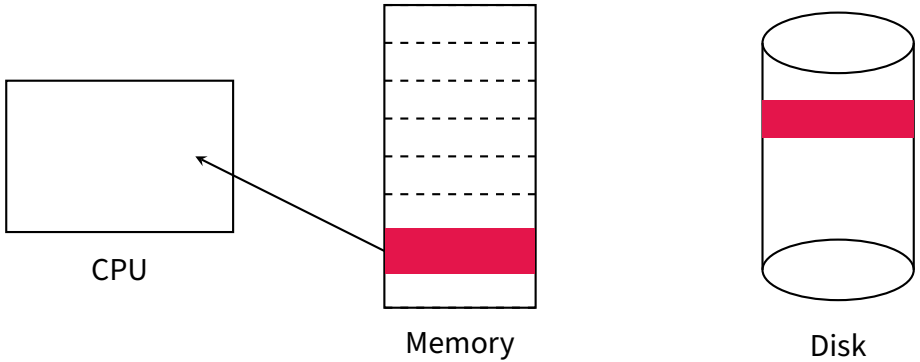


Memory

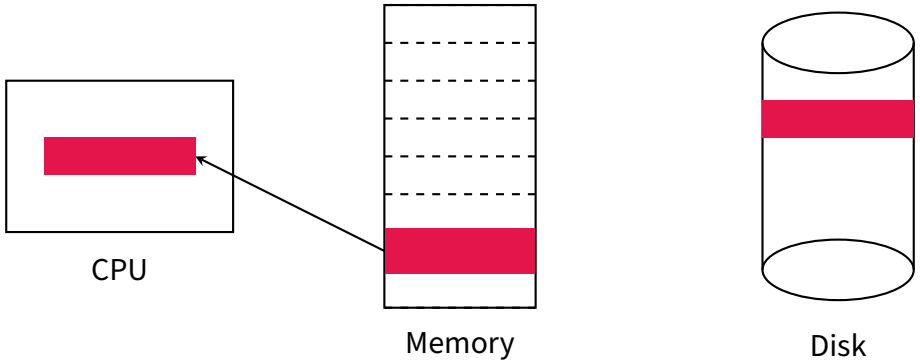


Disk

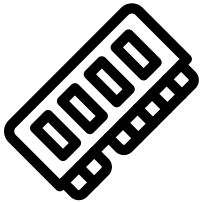
The Page Cache



The Page Cache

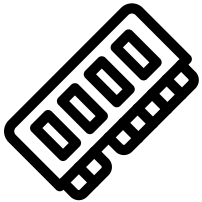


The Linux Page Cache



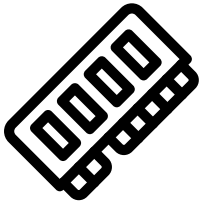
- Operating Systems: Cache pages in memory (RAM)
- Faster Accesses: Shared Libraries, Binaries, Files

The Linux Page Cache



- Operating Systems: Cache pages in memory (RAM)
- Faster Accesses: Shared Libraries, Binaries, Files
- AMD Ryzen 7 7700X [DDR5, Fast SSD]:
  - Disk: 46.6 µs vs. Memory: 0.86 µs
  - 54.2× speedup

The Linux Page Cache



- Operating Systems: Cache pages in memory (RAM)
- Faster Accesses: Shared Libraries, Binaries, Files
- AMD Ryzen 7 7700X [DDR5, Fast SSD]:
  - Disk: 46.6 µs vs. Memory: 0.86 µs
  - 54.2× speedup
- AMD Ryzen 7 PRO 5850U [DDR4, Somewhat Fast SSD]:
  - Disk: 510 µs vs. Memory: 1.06 µs
  - 481.1× speedup

Side-Channel Attacks on the Page Cache

- File $\Longrightarrow$ collection of pages

Side-Channel Attacks on the Page Cache

- File $\Longrightarrow$ collection of pages
- Page is in cache $\Longrightarrow$ page is accessed

Side-Channel Attacks on the Page Cache

- File $\Longrightarrow$ collection of pages
- Page is in cache $\Longrightarrow$ page is accessed
- Page is accessed $\Longrightarrow$ part of file was accessed

Side-Channel Attacks on the Page Cache

- File $\Longrightarrow$ collection of pages
- Page is in cache $\Longrightarrow$ page is accessed
- Page is accessed $\Longrightarrow$ part of file was accessed
- Binary executable $\Longrightarrow$ binary was probably executed

Side-Channel Attacks on the Page Cache

- File $\Longrightarrow$ collection of pages
- Page is in cache $\Longrightarrow$ page is accessed
- Page is accessed $\Longrightarrow$ part of file was accessed
- Binary executable $\Longrightarrow$ binary was probably executed
- Shared library $\Longrightarrow$ code, symbols, data, was probably accessed

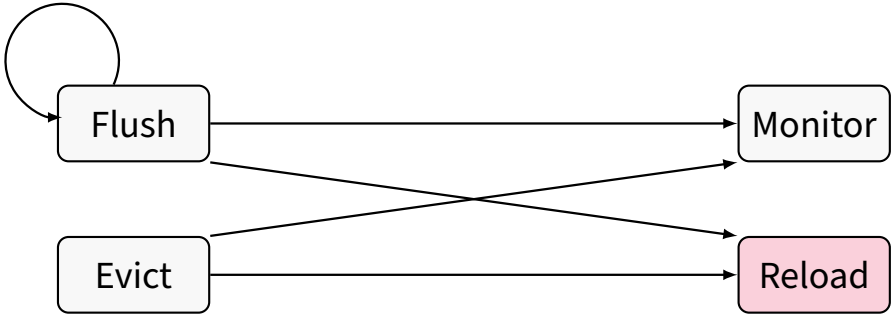
Side-Channel Attacks on the Page Cache

- File $\Longrightarrow$ collection of pages
- Page is in cache $\Longrightarrow$ page is accessed
- Page is accessed $\Longrightarrow$ part of file was accessed
- Binary executable $\Longrightarrow$ binary was probably executed
- Shared library $\Longrightarrow$ code, symbols, data, was probably accessed
- Local Cross-User Threat Model

Side-Channel Attacks on the Page Cache

- File $\Longrightarrow$ collection of pages
- Page is in cache $\Longrightarrow$ page is accessed
- Page is accessed $\Longrightarrow$ part of file was accessed
- Binary executable $\Longrightarrow$ binary was probably executed
- Shared library $\Longrightarrow$ code, symbols, data, was probably accessed
- Local Cross-User Threat Model
- How to interact with the page cache?

The Four Primitives



Reload

- Time taken to *read* a page leaks whether the page is cached

Reload

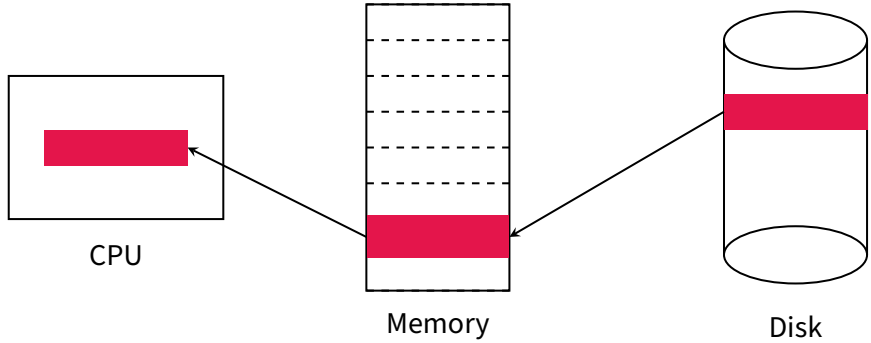
- Time taken to *read* a page leaks whether the page is cached
- Long time $\Longrightarrow$ Disk

Reload

- Time taken to *read* a page leaks whether the page is cached
- Long time $\Longrightarrow$ Disk
- Short time $\Longrightarrow$ Page Cache

Reload

- Time taken to *read* a page leaks whether the page is cached
- Long time $\Longrightarrow$ Disk
- Short time $\Longrightarrow$ Page Cache



Reload

- Manual analysis:

Reload

- Manual analysis: 11 syscalls can reload pages, *i.e.*, timing difference between cache vs disk

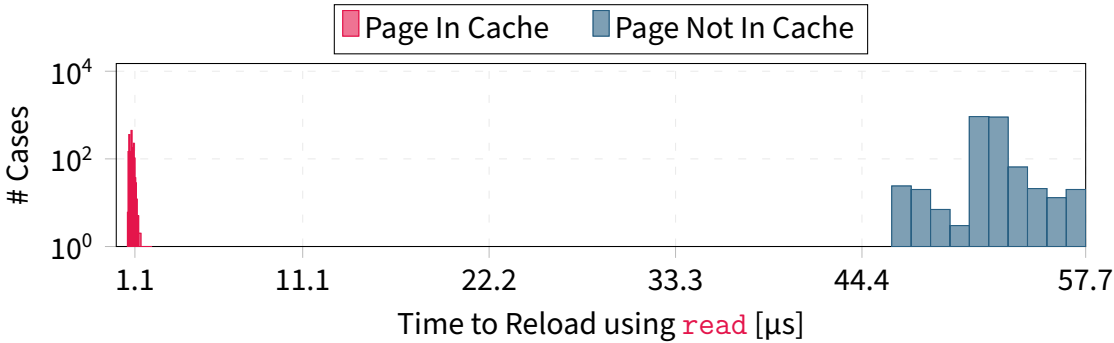
Reload

- Manual analysis: 11 syscalls can reload pages, *i.e.*, timing difference between cache vs disk
- 5.7–147.1× slower than cache

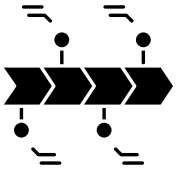
Reload

- Manual analysis: 11 syscalls can reload pages, *i.e.*, timing difference between cache vs disk
- 5.7–147.1× slower than cache
- `copy_file_range`, `mmap` (access), `posix_fadvise` + `POSIX_FADV_WILLNEED`, `pread`, `preadv`, `preadv2`, `read`, `readahead`, `readv`, `sendfile`, `splice`

Reload Timings

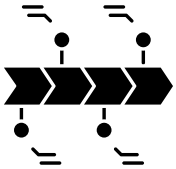


The Readahead Problem



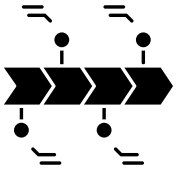
- Readahead: Linux reads 32 pages *ahead* upon access

The Readahead Problem



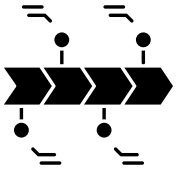
- Readahead: Linux reads 32 pages *ahead* upon access
- Great speed up for normal programs

The Readahead Problem



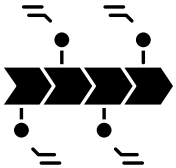
- Readahead: Linux reads 32 pages *ahead* upon access
- Great speed up for normal programs
- Challenge: monitor consecutive pages

The Readahead Problem



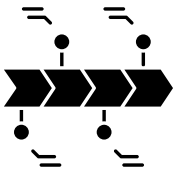
- Readahead: Linux reads 32 pages *ahead* upon access
- Great speed up for normal programs
- Challenge: monitor consecutive pages
- Next page is always pulled in $\Longrightarrow$ Noise

The Readahead Problem



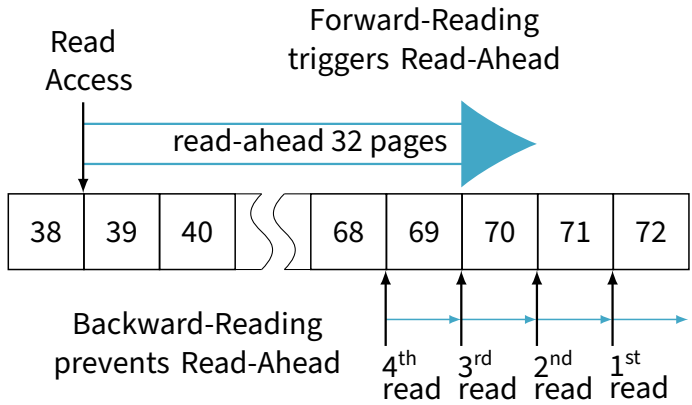
- Readahead: Linux reads 32 pages *ahead* upon access
- Great speed up for normal programs
- Challenge: monitor consecutive pages
- Next page is always pulled in $\Longrightarrow$ Noise
- `MADV_RANDOM`, memory pressure $\Leftarrow$ does not always work

The Readahead Problem



- Readahead: Linux reads 32 pages *ahead* upon access
- Great speed up for normal programs
- Challenge: monitor consecutive pages
- Next page is always pulled in $\Longrightarrow$ Noise
- `MADV_RANDOM`, memory pressure $\Leftarrow$ does not always work
- Our work: 2 novel methods to completely bypass readahead

Backward Reading



Bypassing Readahead: The `readahead` syscall

- The `readahead` syscall

Bypassing Readahead: The `readahead` syscall

- The `readahead` syscall
- `posix_fadvise` syscall + `POSIX_FADV_WILLNEED` flag

Bypassing Readahead: The `readahead` syscall

- The `readahead` syscall
- `posix_fadvise` syscall + `POSIX_FADV_WILLNEED` flag
- Explicit request: pull in exactly specified pages

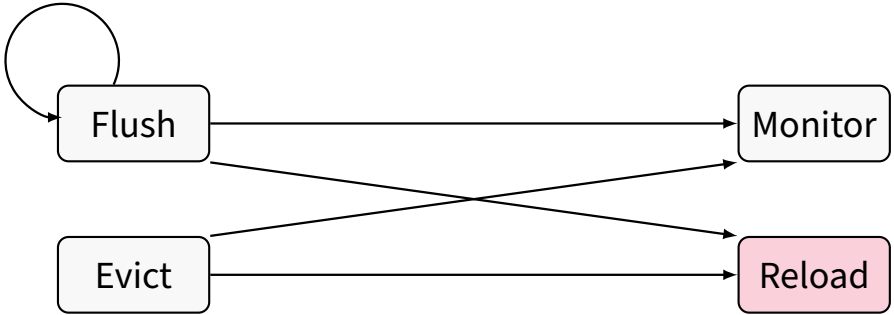
Bypassing Readahead: The `readahead` syscall

- The `readahead` syscall
- `posix_fadvise` syscall + `POSIX_FADV_WILLNEED` flag
- Explicit request: pull in exactly specified pages
- Timing difference: $\sim$0.46 µs (in cache) to $\sim$2.64 µs (on disk)

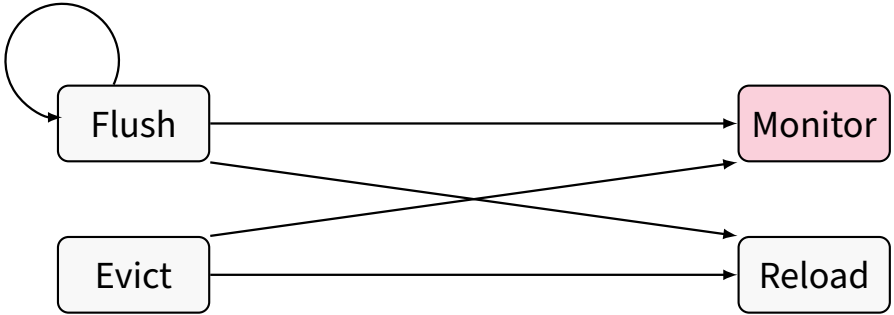
Bypassing Readahead: The `readahead` syscall

- The `readahead` syscall
- `posix_fadvise` syscall + `POSIX_FADV_WILLNEED` flag
- Explicit request: pull in exactly specified pages
- Timing difference: $\sim$0.46 µs (in cache) to $\sim$2.64 µs (on disk)
- Does not trigger the readahead mechanism

The Four Primitives



The Four Primitives



Monitor

- Reload: timing difference $\Longrightarrow$ page cache presence

Monitor

- Reload: timing difference $\Longrightarrow$ page cache presence
- Monitor: directly reports page cache presence

Monitor

- Reload: timing difference $\Longrightarrow$ page cache presence
- Monitor: directly reports page cache presence
- No timing

Monitor

- Reload: timing difference $\Longrightarrow$ page cache presence
- Monitor: directly reports page cache presence
- No timing
- Does not bring to cache, if not in cache

Monitor

- Reload: timing difference $\Longrightarrow$ page cache presence
- Monitor: directly reports page cache presence
- No timing
- Does not bring to cache, if not in cache
- Provided by the operating system

Monitor

- Reload: timing difference $\Longrightarrow$ page cache presence
- Monitor: directly reports page cache presence
- No timing
- Does not bring to cache, if not in cache
- Provided by the operating system

Monitor

- 3 Monitor Syscalls

Monitor

- 3 Monitor Syscalls
- `mincore`: mitigated in 2019 [2, 1]

Monitor

- 3 Monitor Syscalls
- `mincore`: mitigated in 2019 [2, 1]
- `preadv2 + RWF_NOWAIT`: unmitigated [4]

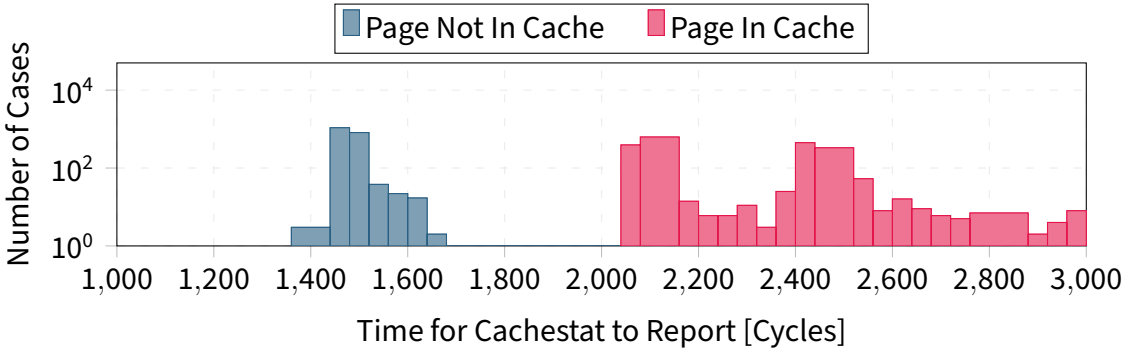
Monitor

- 3 Monitor Syscalls
- `mincore`: mitigated in 2019 [2, 1]
- `preadv2 + RWF_NOWAIT`: unmitigated [4]
- cachestat: mitigated in 2025 after our disclosure [5]

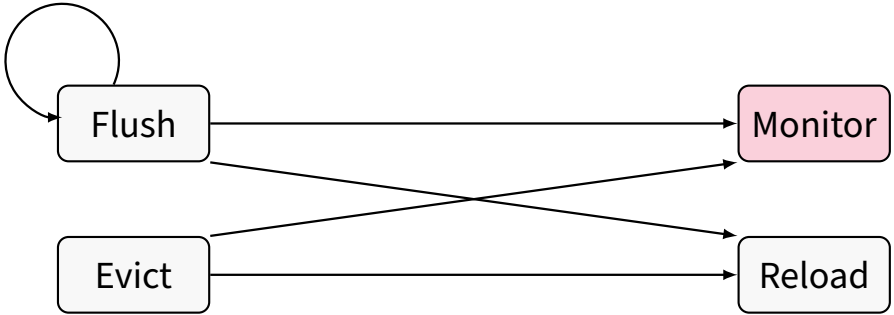
Monitor

- 3 Monitor Syscalls
- `mincore`: mitigated in 2019 [2, 1]
- `preadv2 + RWF_NOWAIT`: unmitigated [4]
- cachestat: mitigated in 2025 after our disclosure [5]
- Blazing fast: reports cache presence of any readable page

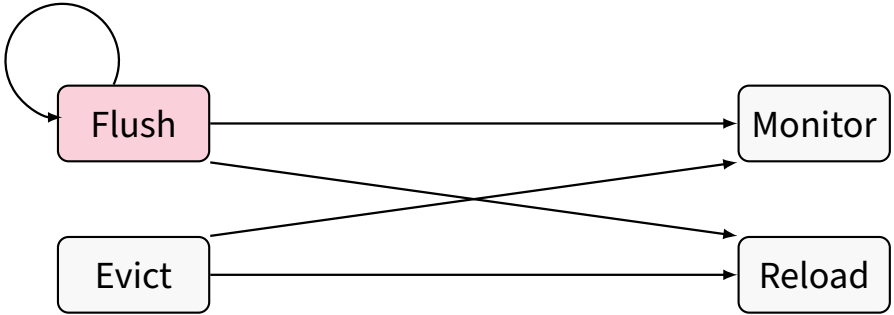
Monitor Timings



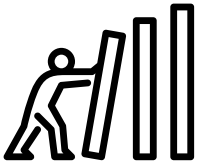
The Four Primitives



The Four Primitives

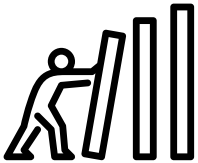


Flush



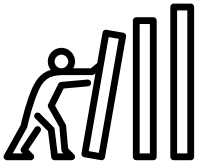
- So far: reload, monitor $\Longrightarrow$ report presence in cache

Flush



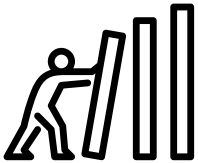
- So far: reload, monitor $\Longrightarrow$ report presence in cache
- Once page in cache: Method to remove from cache

Flush



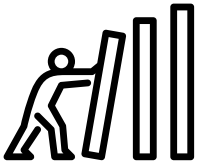
- So far: reload, monitor $\Longrightarrow$ report presence in cache
- Once page in cache: Method to remove from cache
- Flush: Unprivileged, deterministic method to remove pages from cache

Flush



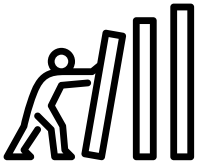
- So far: reload, monitor $\Longrightarrow$ report presence in cache
- Once page in cache: Method to remove from cache
- Flush: Unprivileged, deterministic method to remove pages from cache
- `posix_fadvise` syscall + `POSIX_FADV_DONTNEED` flag

Flush



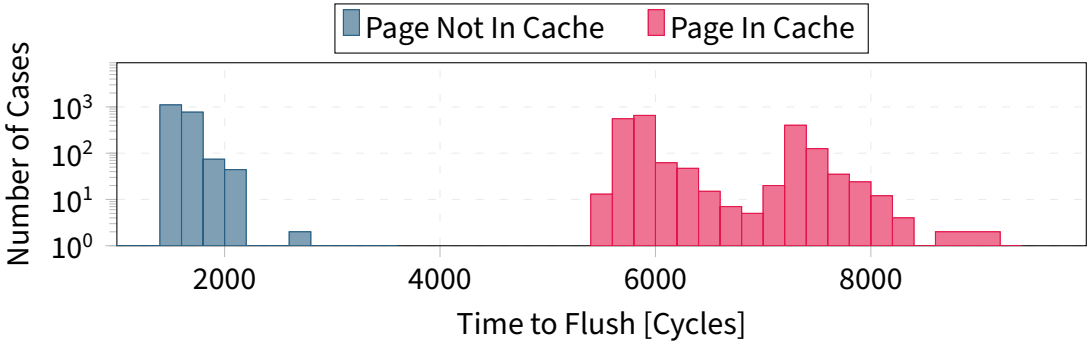
- So far: reload, monitor $\Longrightarrow$ report presence in cache
- Once page in cache: Method to remove from cache
- Flush: Unprivileged, deterministic method to remove pages from cache
- `posix_fadvise` syscall + `POSIX_FADV_DONTNEED` flag
- Mapping pages prevents flushing!

Flush

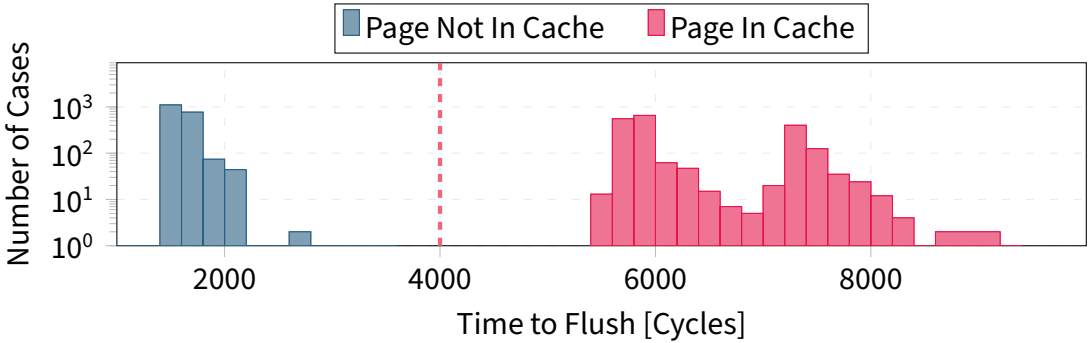


- So far: reload, monitor $\Longrightarrow$ report presence in cache
- Once page in cache: Method to remove from cache
- Flush: Unprivileged, deterministic method to remove pages from cache
- `posix_fadvise` syscall + `POSIX_FADV_DONTNEED` flag
- Mapping pages prevents flushing!
- Speeds up prior work by 6 orders of magnitude

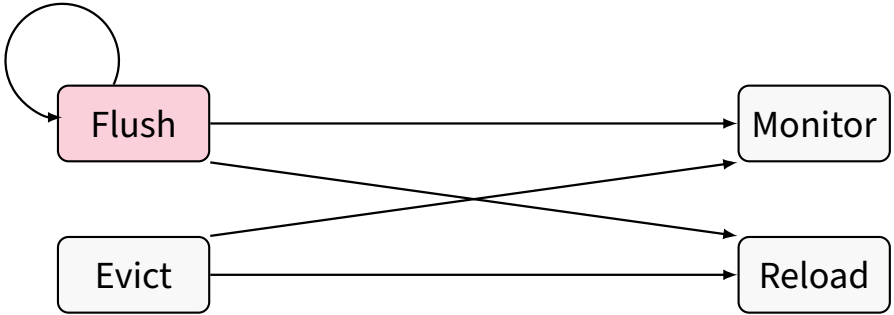
Flush Timings



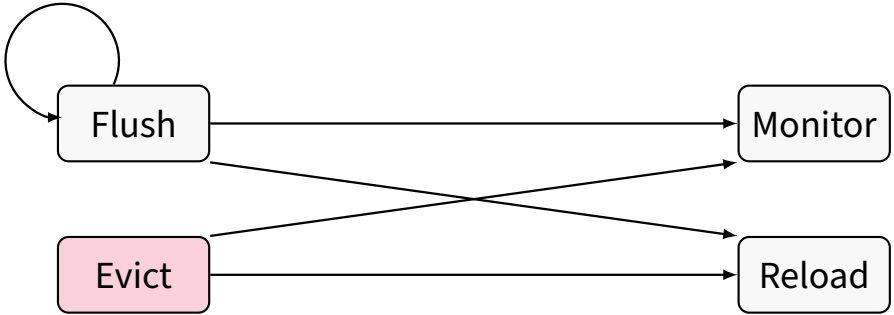
Flush Timings



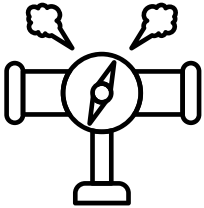
The Four Primitives



The Four Primitives

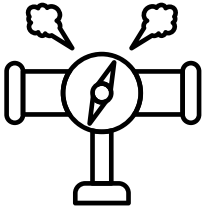


Eviction



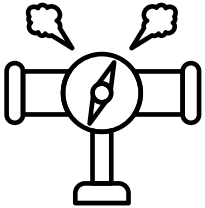
- Flush: Unprivileged, deterministic method to remove page from cache

Eviction



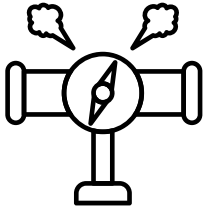
- Flush: Unprivileged, deterministic method to remove page from cache
- Evict: Memory pressure, *i.e.*, fill up RAM $\Longrightarrow$ pages evicted

Eviction



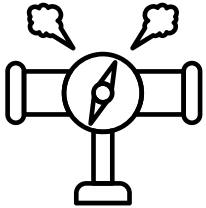
- Flush: Unprivileged, deterministic method to remove page from cache
- Evict: Memory pressure, *i.e.*, fill up RAM $\Longrightarrow$ pages evicted
- Difficult to control, non-deterministic

Eviction



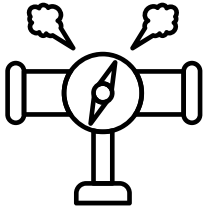
- Flush: Unprivileged, deterministic method to remove page from cache
- Evict: Memory pressure, *i.e.*, fill up RAM $\Longrightarrow$ pages evicted
- Difficult to control, non-deterministic
- Unknown when target pages are evicted

Eviction



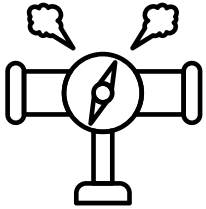
- Flush: Unprivileged, deterministic method to remove page from cache
- Evict: Memory pressure, *i.e.*, fill up RAM $\Longrightarrow$ pages evicted
- Difficult to control, non-deterministic
- Unknown when target pages are evicted
- State of the Art [2]: 3 eviction sets
  - Read useful pages

Eviction



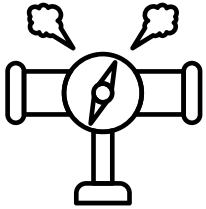
- Flush: Unprivileged, deterministic method to remove page from cache
- Evict: Memory pressure, *i.e.*, fill up RAM $\Longrightarrow$ pages evicted
- Difficult to control, non-deterministic
- Unknown when target pages are evicted
- State of the Art [2]: 3 eviction sets
  - Read useful pages
  - Randomly access uncached pages

Eviction



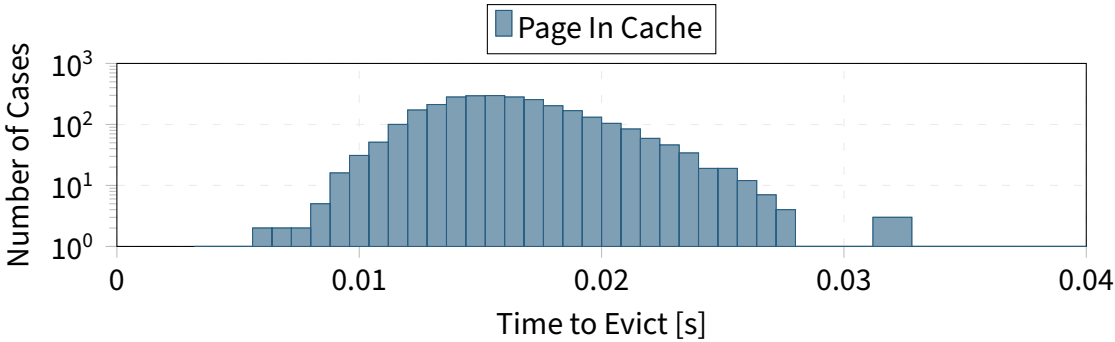
- Flush: Unprivileged, deterministic method to remove page from cache
- Evict: Memory pressure, *i.e.*, fill up RAM $\Longrightarrow$ pages evicted
- Difficult to control, non-deterministic
- Unknown when target pages are evicted
- State of the Art [2]: 3 eviction sets
  - Read useful pages
  - Randomly access uncached pages
  - Baseline pressure (`malloc` then `mlock`)

Eviction



- Flush: Unprivileged, deterministic method to remove page from cache
- Evict: Memory pressure, *i.e.*, fill up RAM $\Longrightarrow$ pages evicted
- Difficult to control, non-deterministic
- Unknown when target pages are evicted
- State of the Art [2]: 3 eviction sets
  - Read useful pages
  - Randomly access uncached pages
  - Baseline pressure (`malloc` then `mlock`)
- Fourth set: Dynamic pressure reacting to memory footprint

Eviction Timings



The Four Primitives

Flush

Monitor

Evict

Reload

The Five Attacks

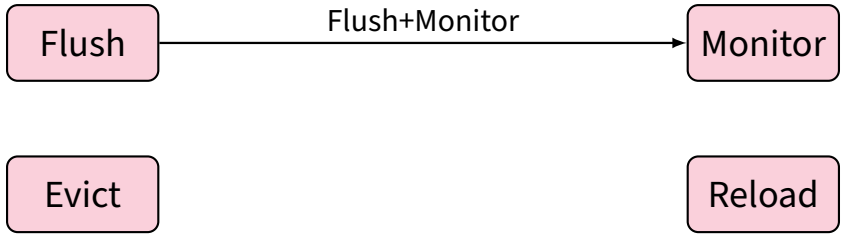
Flush

Monitor

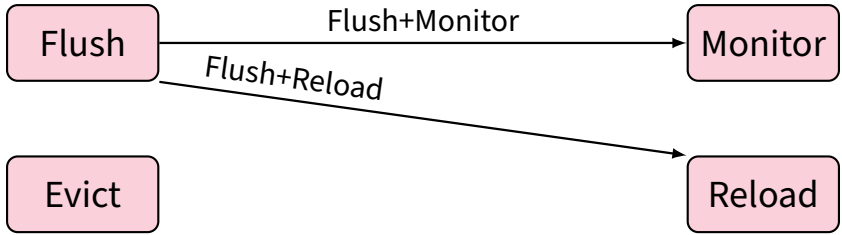
Evict

Reload

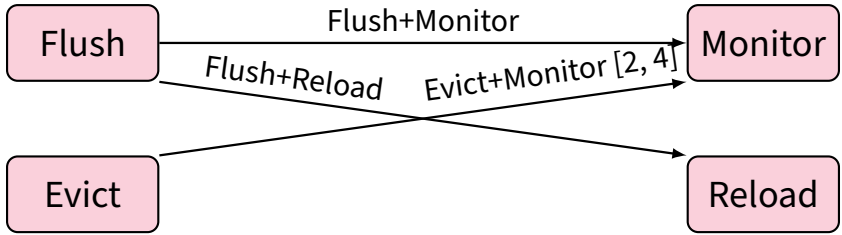
The Five Attacks



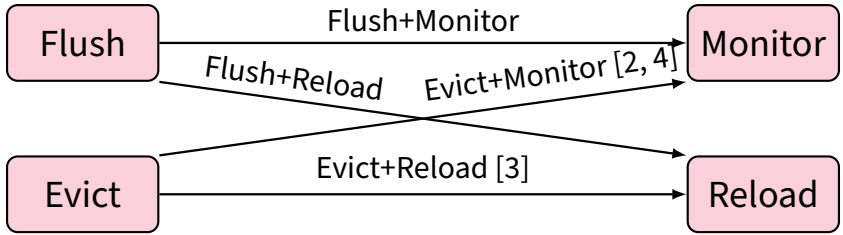
The Five Attacks



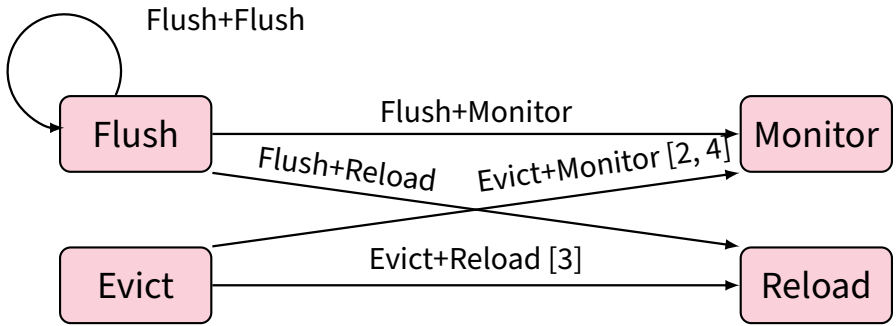
The Five Attacks



The Five Attacks



The Five Attacks



Covert Channels

Attack Technique	Error Rate	Channel Capacity
Flush+Monitor	$< 3 \times 10^{-4}\%$	$37.7\,\mathrm{kB/s}$
Flush+Flush	$1.8\,\%$	$33.9\,\mathrm{kB/s}$
Flush+Reload	$< 2.9 \times 10^{-3}\%$	$4.3\,\mathrm{kB/s}$
Evict+Monitor	$< 8.9 \times 10^{-3}\%$	$1.4\,\mathrm{kB/s}$
Evict+Reload	$6 \times 10^{-4}\%$	$1.2\,\mathrm{kB/s}$

Covert Channel: Hardware and Filesystem

CPU	OS	Kernel	RAM	Capacity
Intel Ultra 7 265K	Ubuntu 24.04	6.14.0	8 GB	73.1 kB/s
Intel i9 13900KF	Ubuntu 22.04	6.8.0	64 GB	71.5 kB/s
Intel i5-11300H	Debian 13	6.12.43	16 GB	55.0 kB/s
Intel i5-8265U	Ubuntu 22.04	6.8.0	16 GB	35.9 kB/s
AMD Epyc 7313P	Ubuntu 22.04	6.11.0	64 GB	31.8 kB/s

Flush+Monitor Across Hardware

Covert Channel: Hardware and Filesystem

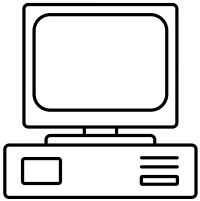
CPU	OS	Kernel	RAM	Capacity
Intel Ultra 7 265K	Ubuntu 24.04	6.14.0	8 GB	73.1 kB/s
Intel i9 13900KF	Ubuntu 22.04	6.8.0	64 GB	71.5 kB/s
Intel i5-11300H	Debian 13	6.12.43	16 GB	55.0 kB/s
Intel i5-8265U	Ubuntu 22.04	6.8.0	16 GB	35.9 kB/s
AMD Epyc 7313P	Ubuntu 22.04	6.11.0	64 GB	31.8 kB/s

Flush+Monitor Across Hardware

File System	btrfs	ext4	ext3	ext2	ntfs
Channel Capacity	64.3 kB/s	54.99 kB/s	43.8 kB/s	44.4 kB/s	48.3 kB/s

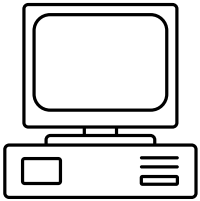
Flush+Monitor Across Filesystems

Covert Channels



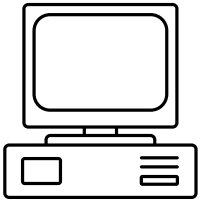
- Intel 7 Ultra 155U, Ubuntu 18.04.6, kernel 5.4.0-84-generic

Covert Channels



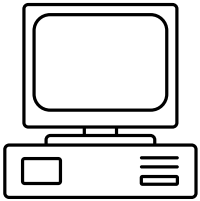
- Intel 7 Ultra 155U, Ubuntu 18.04.6, kernel 5.4.0-84-generic
  - Flush+Monitor (`preadv2`): 69.1 $\mathrm{kB/s}$
  - Flush+Reload: 5.1 $\mathrm{kB/s}$

Covert Channels



- Intel 7 Ultra 155U, Ubuntu 18.04.6, kernel 5.4.0-84-generic
  - Flush+Monitor (`preadv2`): 69.1 $\mathrm{kB/s}$
  - Flush+Reload: 5.1 $\mathrm{kB/s}$
- Intel i5-11300H, Debian 12, kernel 6.1.0-39

Covert Channels



- Intel 7 Ultra 155U, Ubuntu 18.04.6, kernel 5.4.0-84-generic
  - Flush+Monitor (`preadv2`): 69.1 $\mathrm{kB/s}$
  - Flush+Reload: 5.1 $\mathrm{kB/s}$
- Intel i5-11300H, Debian 12, kernel 6.1.0-39
  - Flush+Monitor (`preadv2`): 37.3 $\mathrm{kB/s}$
  - Flush+Reload: 2.68 $\mathrm{kB/s}$

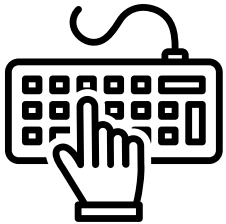
Inter-Keystroke Timing

- Evict+Monitor on Gedit: Page 32 of `libgedit-41.so`

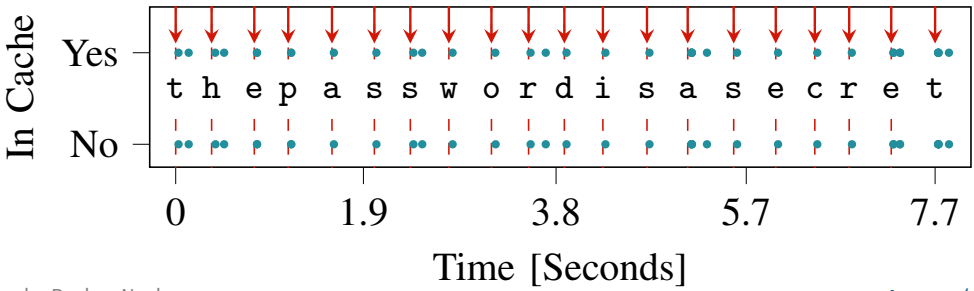
Inter-Keystroke Timing

- Evict+Monitor on Gedit: Page 32 of `libgedit-41.so`
- Accessed upon any keypress (monitor), removed after keypress (evict);

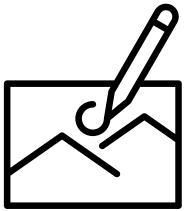
Inter-Keystroke Timing



- Evict+Monitor on Gedit: Page 32 of `libgedit-41.so`
- Accessed upon any keypress (monitor), removed after keypress (evict); cannot flush!
- 5.5 keys/s, F_1 score: 96.7%

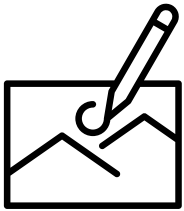


Authentication UI Redress



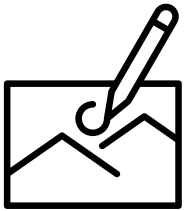
- Flush+Monitor: detect when `pkexec` is run

Authentication UI Redress



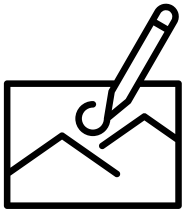
- Flush+Monitor: detect when `pkexec` is run
- `pkexec`: Spawns UI password prompt

Authentication UI Redress



- Flush+Monitor: detect when `pkexec` is run
- `pkexec`: Spawns UI password prompt
- Draw a fake password prompt over the real one

Authentication UI Redress



- Flush+Monitor: detect when `pkexec` is run
- `pkexec`: Spawns UI password prompt
- Draw a fake password prompt over the real one
- Detect the first page of pkexec in 800 ns

Cross-Container

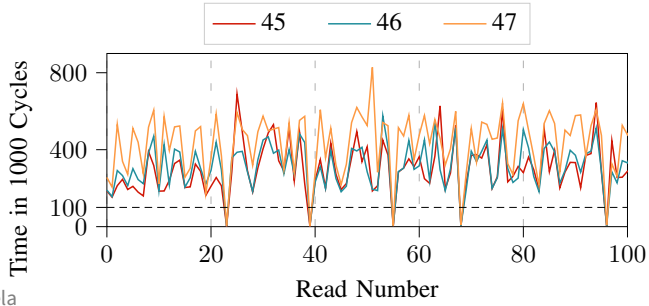
- Across containers of the same image

Cross-Container

- Across containers of the same image
- Flush+Reload: detect when shared pages are accessed (OverlayFS)

Cross-Container

- Across containers of the same image
- Flush+Reload: detect when shared pages are accessed (OverlayFS)
- Pages 45-47 of libtinfo.so: When a shell is launched in another container



Website Fingerprinting

- Firefox 133.0: libxul.so: 41,205 pages! Filtering: 7198 pages

Website Fingerprinting

- Firefox 133.0: libxul.so: 41,205 pages! Filtering: 7198 pages
- Top-100 Closed-World evaluation

Website Fingerprinting

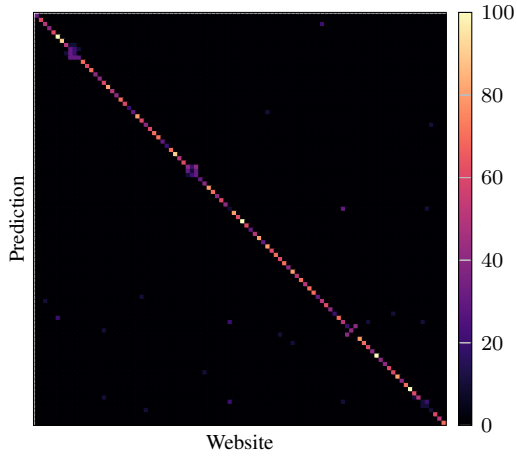
- Firefox 133.0: libxul.so: 41,205 pages! Filtering: 7198 pages
- Top-100 Closed-World evaluation
- 30 traces per website, 10 seconds per trace

Website Fingerprinting

- Firefox 133.0: libxul.so: 41,205 pages! Filtering: 7198 pages
- Top-100 Closed-World evaluation
- 30 traces per website, 10 seconds per trace
- Random Forest classifier

Website Fingerprinting F_1 Scores

Attack Technique	F_1 Score
Flush+Monitor	90.5 %
Flush+Reload	88.1 %
Flush+Flush	86.1 %
Evict+Monitor	79.5 %
Evict+Reload	37.9 %



Responsible Disclosure & What Remains

- The `cachestat` syscall: mitigated in 2025 [5]

Responsible Disclosure & What Remains

- The `cachestat` syscall: mitigated in 2025 [5]
- CVE-2025-21691: Red Hat CVSS v3 7.1, SUSE v4 6.8

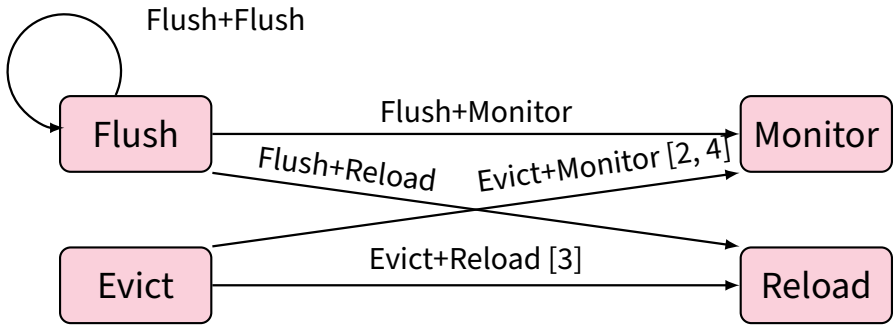
Responsible Disclosure & What Remains

- The `cachestat` syscall: mitigated in 2025 [5]
- CVE-2025-21691: Red Hat CVSS v3 7.1, SUSE v4 6.8
- Only fixes one path of Flush+Monitor, Evict+Monitor

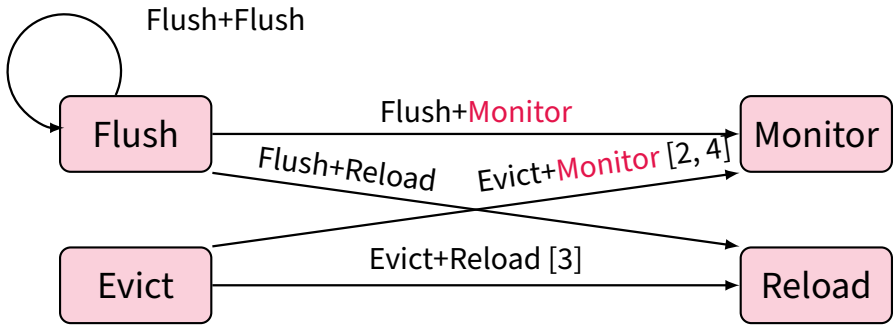
Responsible Disclosure & What Remains

- The `cachestat` syscall: mitigated in 2025 [5]
- CVE-2025-21691: Red Hat CVSS v3 7.1, SUSE v4 6.8
- Only fixes one path of Flush+Monitor, Evict+Monitor

The Five Attacks



The Five Attacks



Conclusion

- In this work: revived and advanced page cache attacks (thought to be mitigated or impractical since 2019)
- Flush, Reload, Evict, Monitor: combine into 5 attacks
- Keystroke Detection, Covert Channels, Website Fingerprinting, Cross-Container Leaks
- Attacks remain possible

sudheendra.neela@tugraz.at
Website: https://snee.la
Mastodon: @vmcall@infosec.exchange
Bluesky: @snee.la
Available, Functional, Reproducible:
https://github.com/isec-tugraz/Eviction-Notice

Acknowledgments

This research was made possible by generous funding from:

Supported in part by the European Research Council (ERC project FSSec 101076409) and the Austrian Science Fund (FWF SFB project SPyCoDe 10.55776/F85 and FWF project 10.55776/I6054) and the Austrian Research Promotion Agency (FFG) via the SEIZE project (FFG grant numbers 888087. Additional funding was provided by generous gifts from Red Hat, Google, and Intel. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding parties.

Bibliography

[1] Novak Boskov et al. **Union Buster: A Cross-Container Covert-Channel Exploiting Union Mounting**. International Symposium on Cyber Security, Cryptology, and Machine Learning. 2022.

[2] Daniel Gruss et al. **Page Cache Attacks**. CCS. 2019.

[3] Jonas Juffinger et al. **The HMB Timing Side Channel: Exploiting the SSD's Host Memory Buffer**. DIMVA. 2025.

[4] Martin Schwarzl, Erik Kraft, and Daniel Gruss. **Layered Binary Templating**. ACNS. 2023.

[5] Linus Torvalds. **cachestat: fix page cache statistics permission checking**. 2025. URL: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=5f537664e705b0bf8b7e329861f20128534f6a83.