TU Graz

SCIENCE
PASSION
TECHNOLOGY

Security Implications of Linux Kernel Electric Fence

Studying and Attacking KFENCE

Ernesto Martínez García

Linux Security Summit Europe 2026

› isec.tugraz.at

$ whoami



Ernesto Martínez García

PhD Student @ Graz University of Technology

ernesto.martinezgarcia@tugraz.at

ernestomartinezgarcia.com

linkedin.com/in/ecomaikgolf

Background



Kernel Exploitation

Kernel Exploitation

> Attackers exploit bugs in the kernel to escalate privileges

Kernel Exploitation

> Attackers exploit bugs in the kernel to escalate privileges

As with userspace exploitation, **allocator exploitation** gained traction

↳ Attackers may solely exploit allocated memory (**data-only** attacks)

Kernel Exploitation

> Attackers exploit bugs in the kernel to escalate privileges

As with userspace exploitation, **allocator exploitation** gained traction

↳ Attackers may solely exploit allocated memory (**data-only** attacks)

Linux allocator hardening is crucial nowadays

↳ Defense-in-depth against CVEs

Kernel Exploitation

> Attackers exploit bugs in the kernel to escalate privileges

As with userspace exploitation, **allocator exploitation** gained traction

↳ Attackers may solely exploit allocated memory (**data-only** attacks)

Linux allocator hardening is crucial nowadays

↳ Defense-in-depth against CVEs

Threat Model

- Unprivileged userspace command execution
- An attacker found an UAF on a kernel object
- We assume a hardened configuration (`hardening.config`)

Slab Hardening

Slab Hardening

> Slab (SLUB) is the core Linux Kernel heap **memory allocator**

Slab Hardening

> Slab (SLUB) is the core Linux Kernel heap **memory allocator**

Contains comprehensive defense-in-depth mechanisms against exploitation

Slab Hardening

> Slab (SLUB) is the core Linux Kernel heap **memory allocator**

Contains comprehensive defense-in-depth mechanisms against exploitation

D1: Heap Segregation

`SLAB_BUCKETS`

`SLAB_VIRTUAL`

`TYPED_KMALLOC_CACHES`

D2: Sanitization

`{KASAN,KMSAN}_*`

`KASAN_HW_TAGS`

`INIT_ON_{ALLOC,FREE}`

D3: Randomization(s)

`RANDOM_KMALLOC_CACHES`[D1]

`SLAB_FREELIST_RANDOM`

D4: Other

`SAMSUNG RKP`

`HUAWEI HYPERVISOR`

`OTHER DOWNSTREAM`

Slab Hardening

> Slab (SLUB) is the core Linux Kernel heap **memory allocator**

Contains comprehensive defense-in-depth mechanisms against exploitation

D1: Heap Segregation

`SLAB_BUCKETS`

`SLAB_VIRTUAL`

`TYPED_KMALLOC_CACHES`

D2: Sanitization

`{KASAN,KMSAN}_*`

`KASAN_HW_TAGS`

`INIT_ON_{ALLOC,FREE}`

D3: Randomization(s)

`RANDOM_KMALLOC_CACHES`[D1]

`SLAB_FREELIST_RANDOM`

Most importantly, **objects are segregated**:

D4: Other

`SAMSUNG RKP`

`HUAWEI HYPERVISOR`

`OTHER DOWNSTREAM`

Slab Hardening

> Slab (SLUB) is the core Linux Kernel heap **memory allocator**

Contains comprehensive defense-in-depth mechanisms against exploitation

D1: Heap Segregation

`SLAB_BUCKETS`

`SLAB_VIRTUAL`

`TYPED_KMALLOC_CACHES`

D2: Sanitization

`{KASAN,KMSAN}_*`

`KASAN_HW_TAGS`

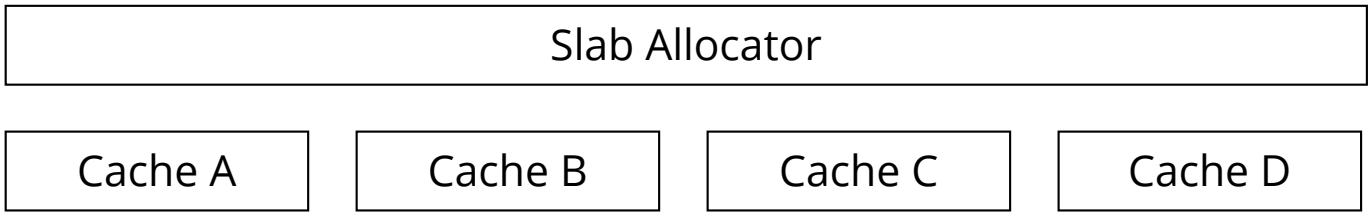
`INIT_ON_{ALLOC,FREE}`

D3: Randomization(s)

`RANDOM_KMALLOC_CACHES`[D1]

`SLAB_FREELIST_RANDOM`

Most importantly, **objects are segregated**:



D4: Other

`SAMSUNG RKP`

`HUAWEI HYPERVISOR`

`OTHER DOWNSTREAM`

Slab Hardening

> Slab (SLUB) is the core Linux Kernel heap **memory allocator**

Contains comprehensive defense-in-depth mechanisms against exploitation

D1: Heap Segregation

SLAB_BUCKETS

SLAB_VIRTUAL

TYPED_KMALLOC_CACHES

D2: Sanitization

{KASAN,KMSAN}_*

KASAN_HW_TAGS

INIT_ON_{ALLOC,FREE}

D3: Randomization(s)

RANDOM_KMALLOC_CACHES[D1]

SLAB_FREELIST_RANDOM

Most importantly, **objects are segregated**:

```
1 free(vuln_sketchy_object)        // -> UAF Weak Code Object
2 very_important_object = alloc() // -> Even With Same Size
3 write(vuln_sketchy_object)       // -> No UAF Memory Overlap
```

D4: Other

SAMSUNG RKP

HUAWEI HYPERVISOR

OTHER DOWNSTREAM

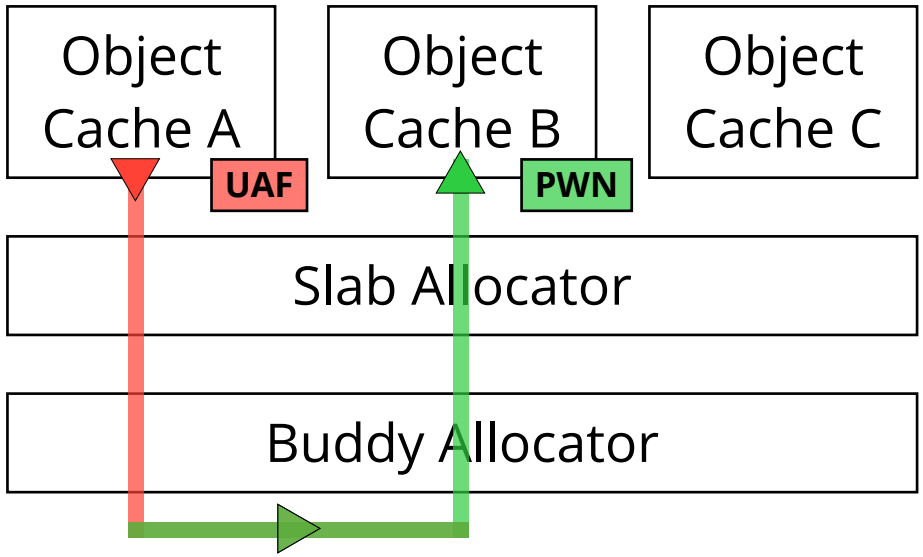
Cross-cache Attacks

Cross-cache Attacks

> Technique to reclaim cross-domain memory via the physical allocator

Cross-cache Attacks

Technique to reclaim cross-domain memory via the physical allocator

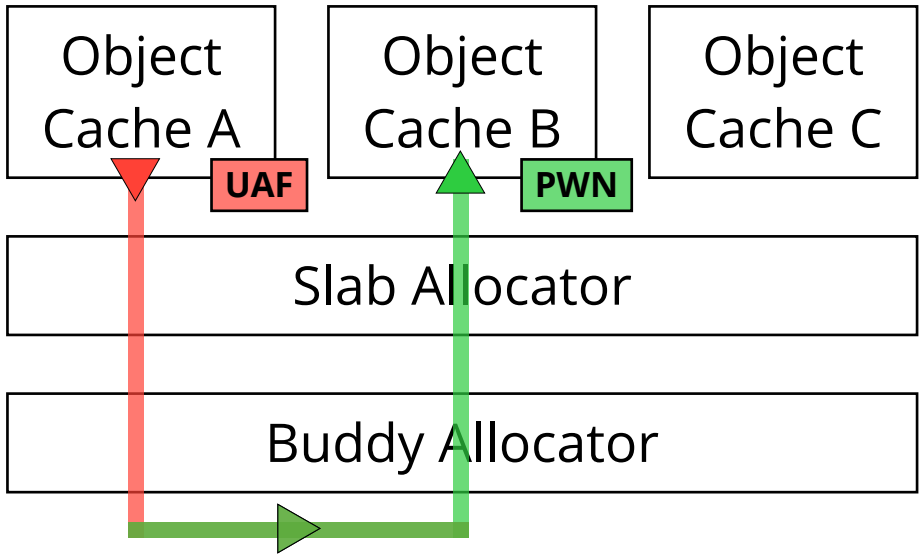


Cross-cache Attacks

> Technique to reclaim cross-domain memory via the physical allocator

Apply **free pressure** on Cache A

↳ Page(s) containing old A are freed



Cross-cache Attacks

> Technique to reclaim cross-domain memory via the physical allocator

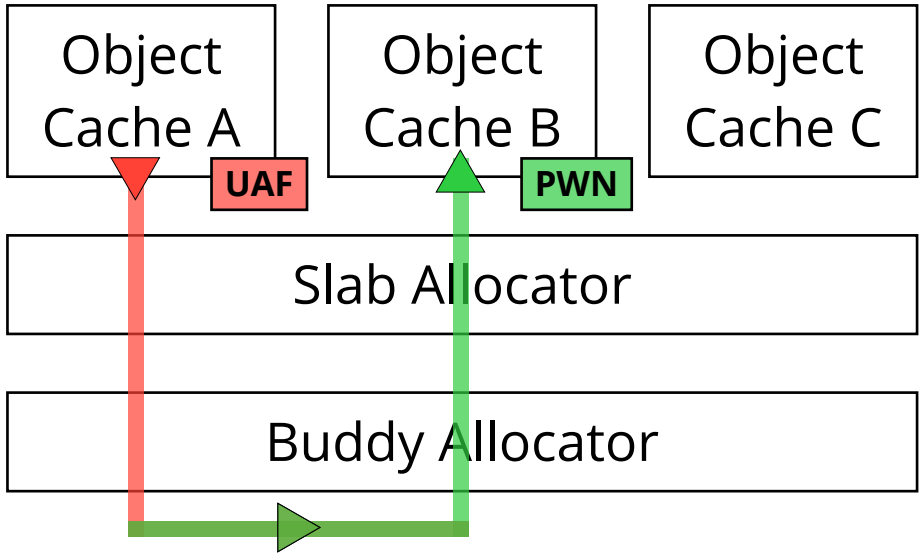
Apply **free pressure** on Cache A

↳ Page(s) containing old A are freed

Apply **alloc pressure** on Cache B

↳ Page is re-allocated for new B

↳ UAF moved from A to B



Cross-cache Attacks

> Technique to reclaim cross-domain memory via the physical allocator

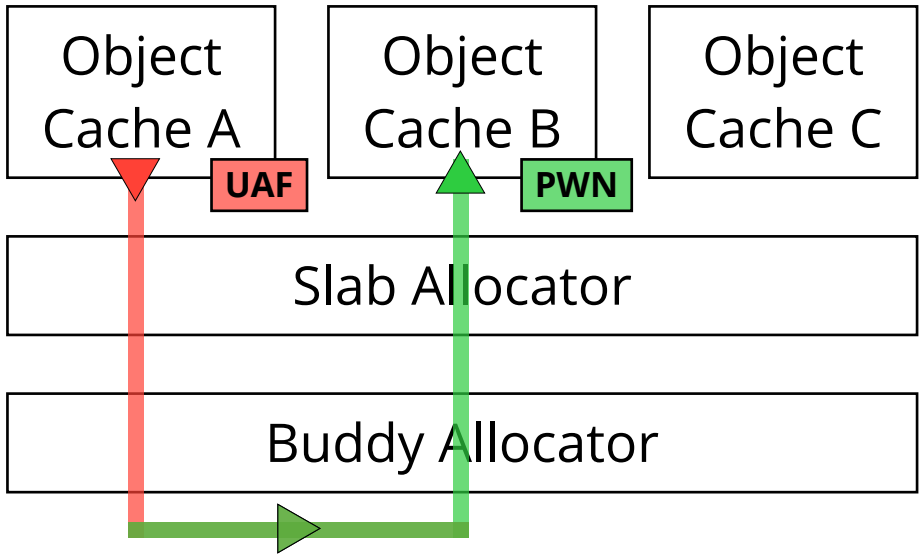
Apply **free pressure** on Cache A

↳ Page(s) containing old A are freed

Apply **alloc pressure** on Cache B

↳ Page is re-allocated for new B

↳ UAF moved from A to B



Future of Cross-cache Attacks

- `SLAB_VIRTUAL` deterministically mitigates them (UAF) — Not Deployed
- Memory Tagging probabillistically defends against them — In Deployment

Kernel Electric Fence



Kernel Electric Fence

Kernel Electric Fence

> A sample-based memory safety error detector for **production kernels**.

Kernel Electric Fence

> A sample-based memory safety error detector for **production kernels**.

Developed by Marco Elver and Alexander Potapenko (Google) in 2020-21 (5.12)

Kernel Electric Fence

> A sample-based memory safety error detector for **production kernels**.

Developed by Marco Elver and Alexander Potapenko (Google) in 2020-21 (5.12)

- Follows a similar design as userspace's **GWP-ASan**
  - ↳ Small sample of allocations are served with sanitizer instrumentation

Kernel Electric Fence

> A sample-based memory safety error detector for **production kernels**.

Developed by Marco Elver and Alexander Potapenko (Google) in 2020-21 (5.12)

- Follows a similar design as userspace's **GWP-ASan**
  - ↳ Small sample of allocations are served with sanitizer instrumentation

Catches complicated bugs on actual production systems

Kernel Electric Fence

> A sample-based memory safety error detector for **production kernels**.

Developed by Marco Elver and Alexander Potapenko (Google) in 2020-21 (5.12)

💡 Follows a similar design as userspace's **GWP-ASan**

↳ Small sample of allocations are served with sanitizer instrumentation

Catches complicated bugs on actual production systems

Properties

- Near zero overhead
- Probabilistic approach
- No complicated setup

Kernel Electric Fence

> A sample-based memory safety error detector for **production kernels**.

Developed by Marco Elver and Alexander Potapenko (Google) in 2020-21 (5.12)

💡 Follows a similar design as userspace's **GWP-ASan**

↳ Small sample of allocations are served with sanitizer instrumentation

Catches complicated bugs on actual production systems

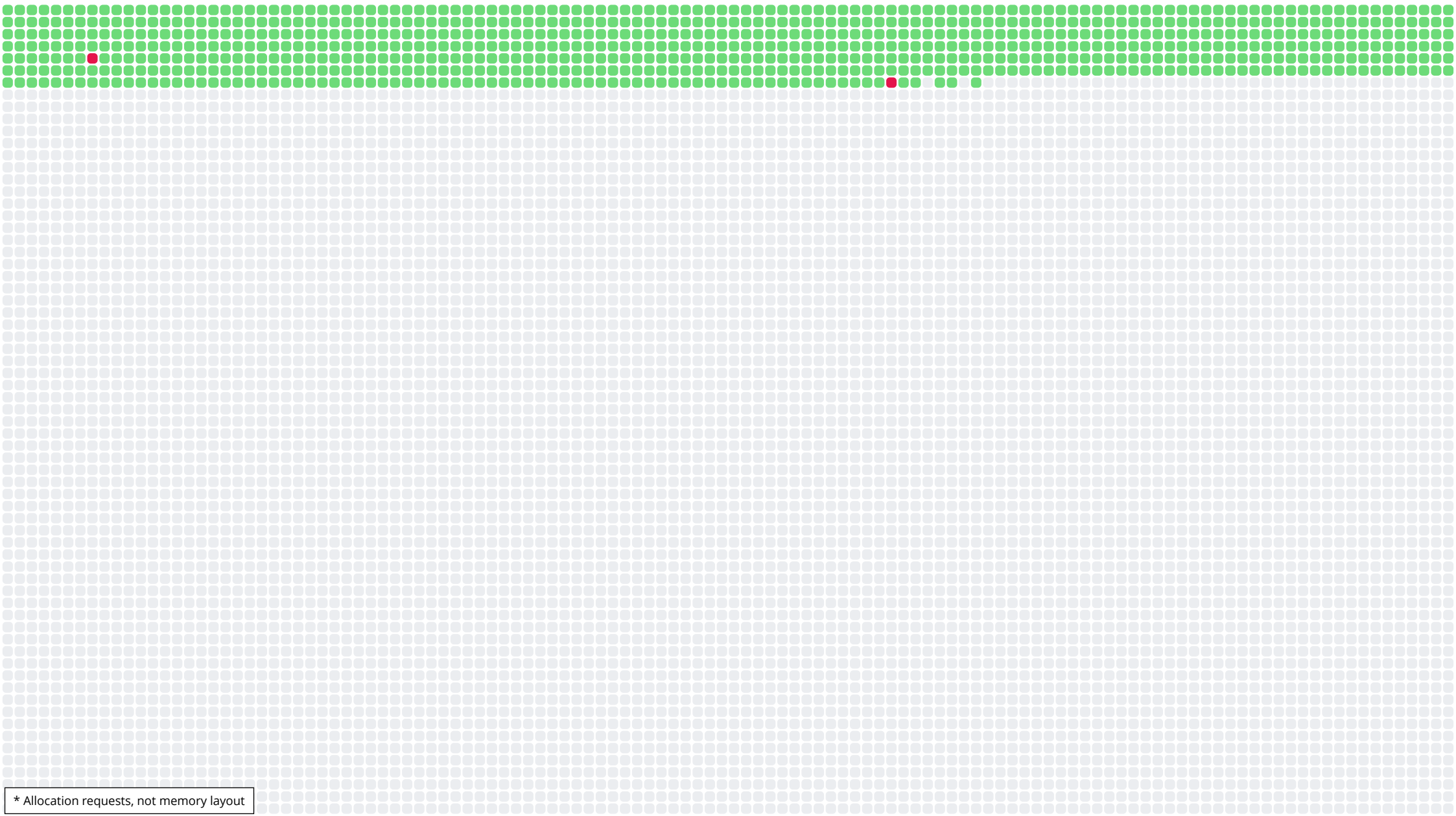
Properties

- Near zero overhead
- Probabilistic approach
- No complicated setup

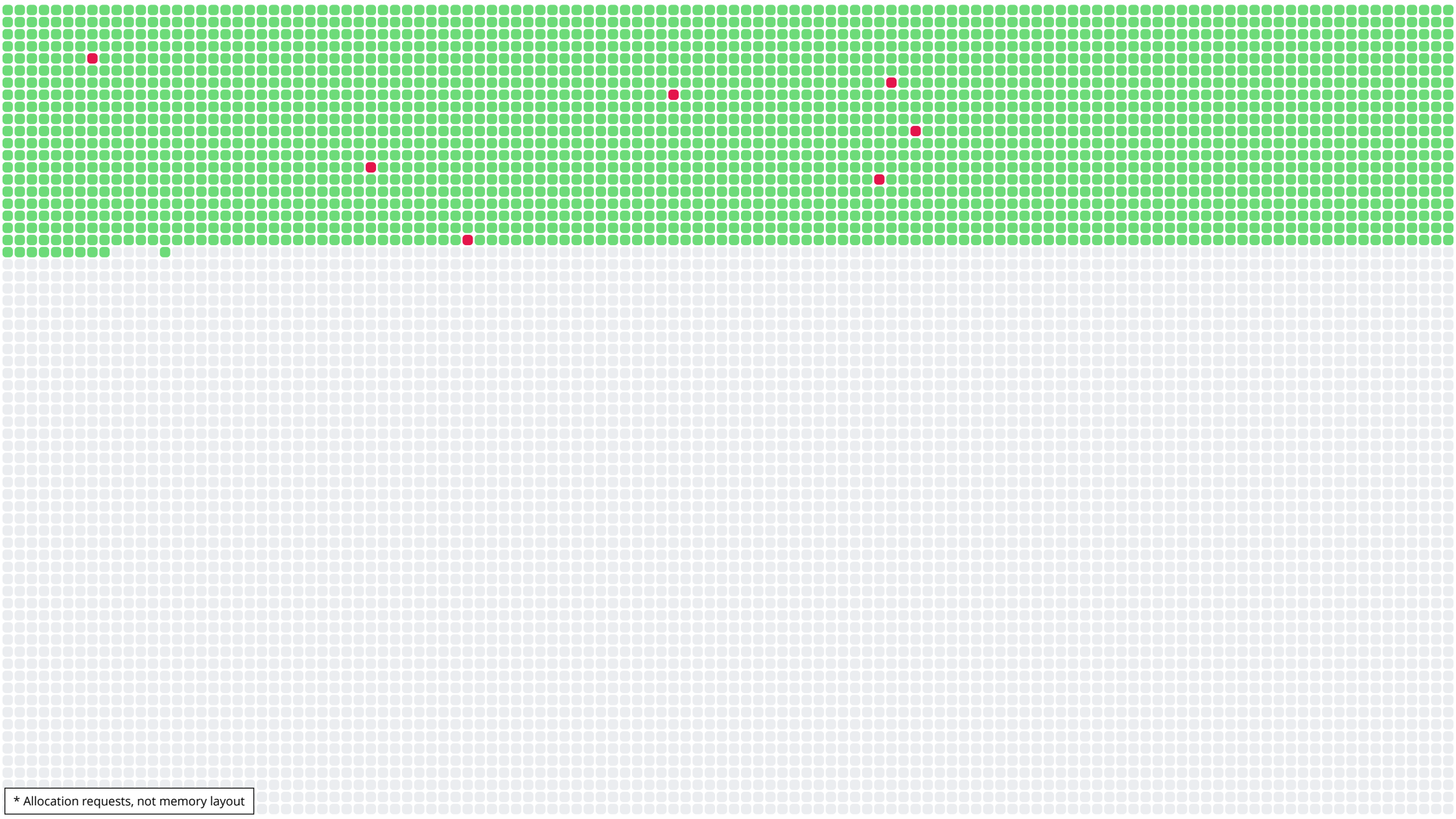
Default Enabled



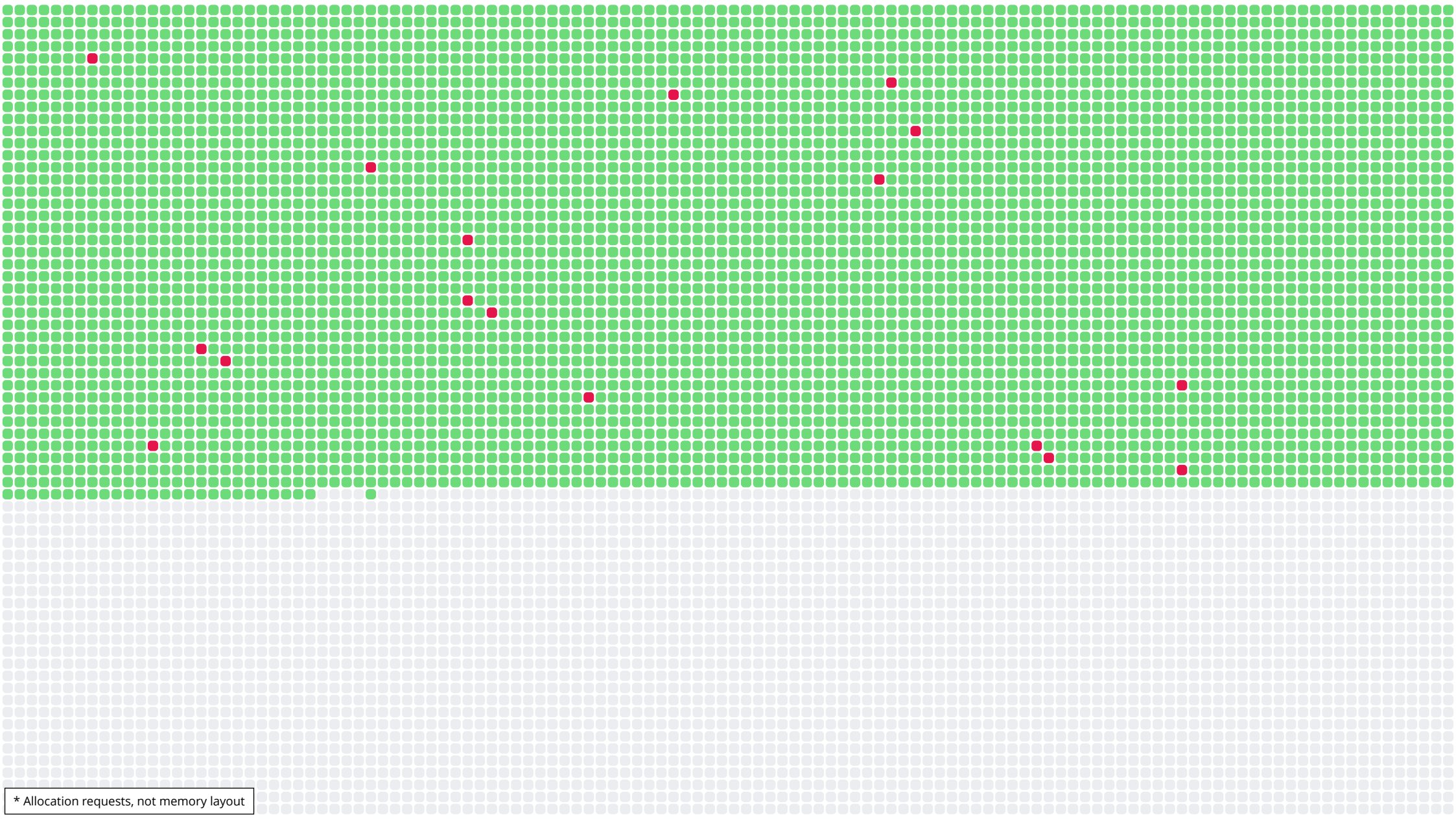
* Allocation requests, not memory layout



* Allocation requests, not memory layout



* Allocation requests, not memory layout

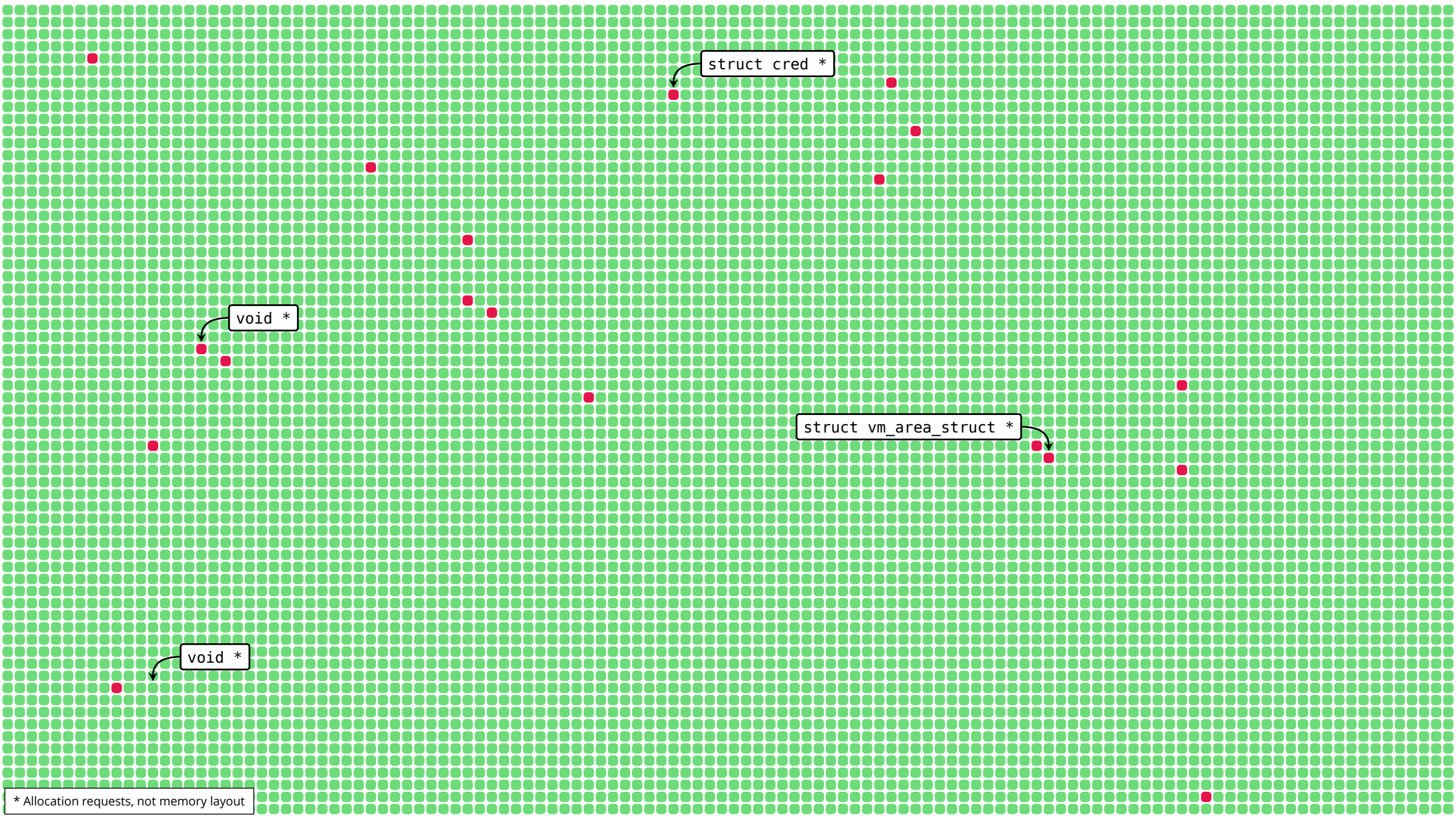


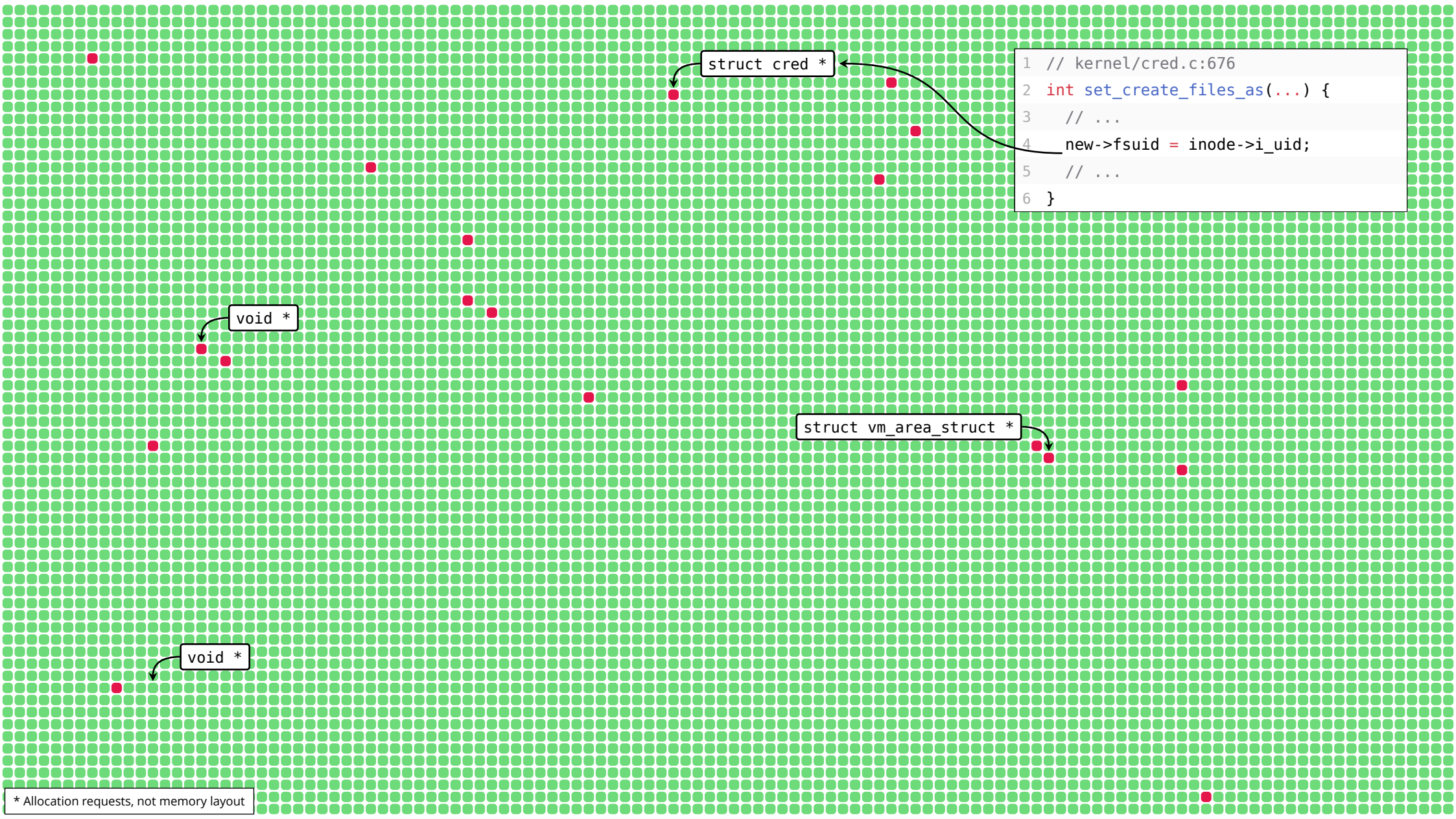
* Allocation requests, not memory layout

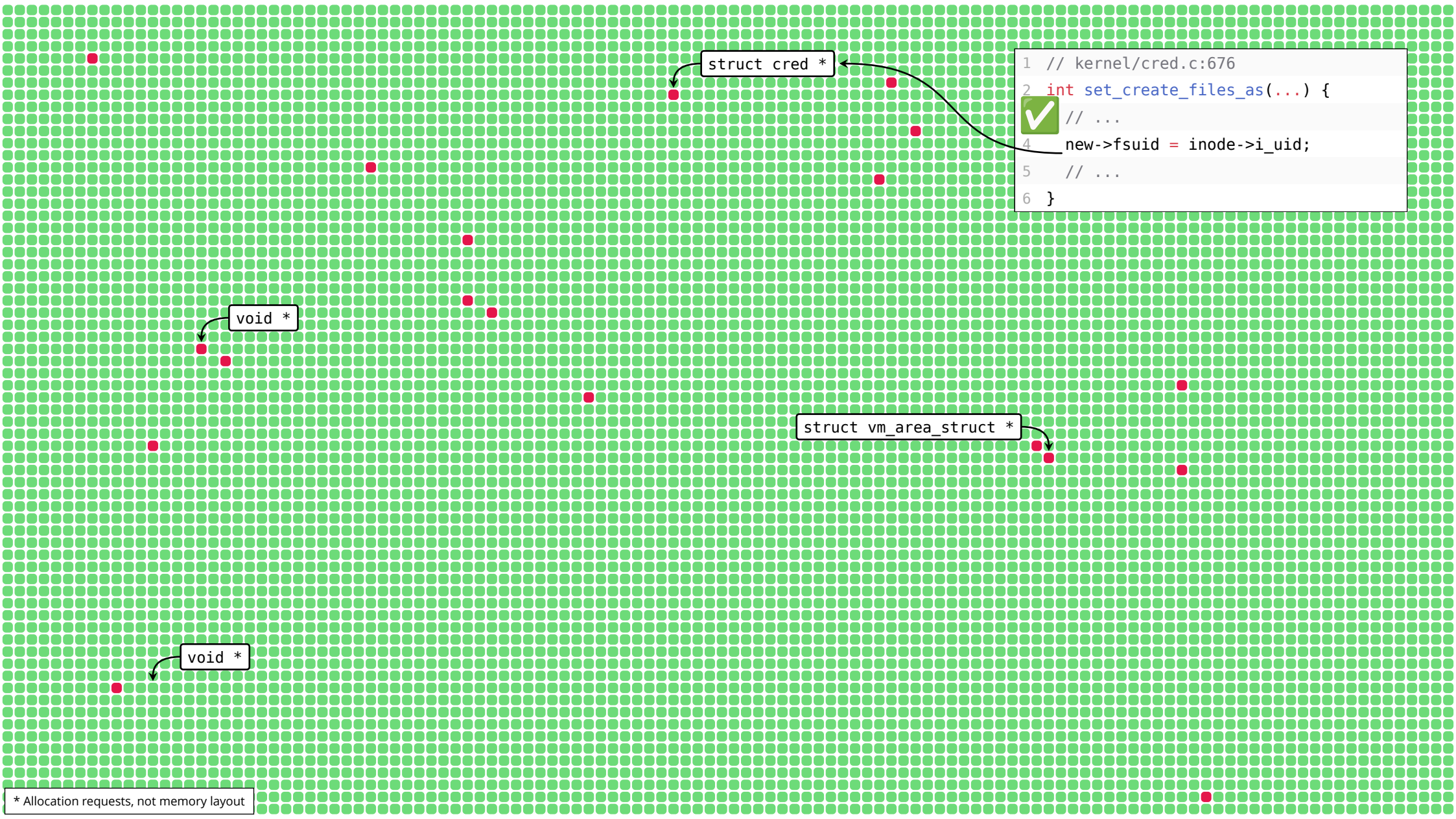


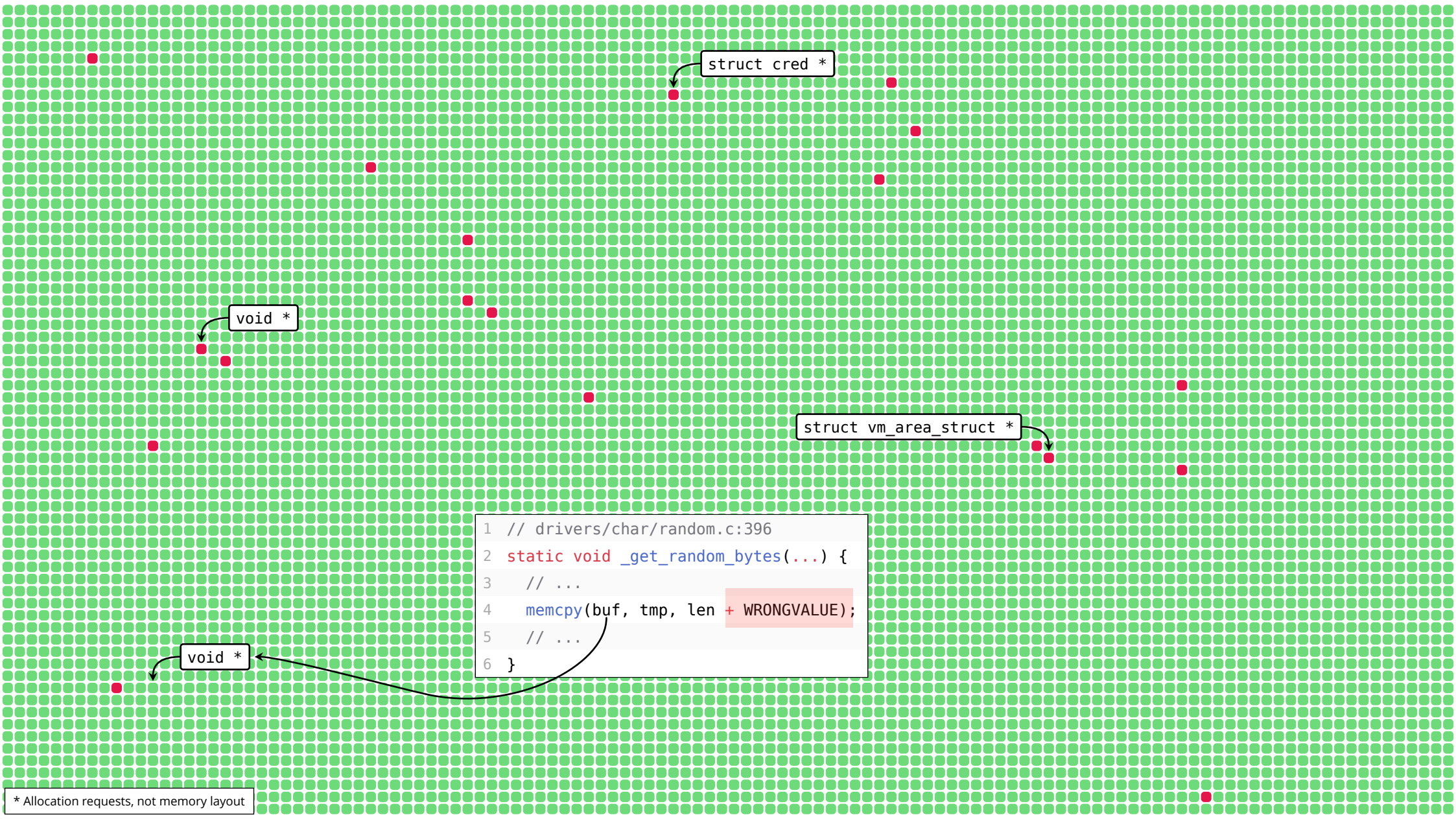
* Allocation requests, not memory layout

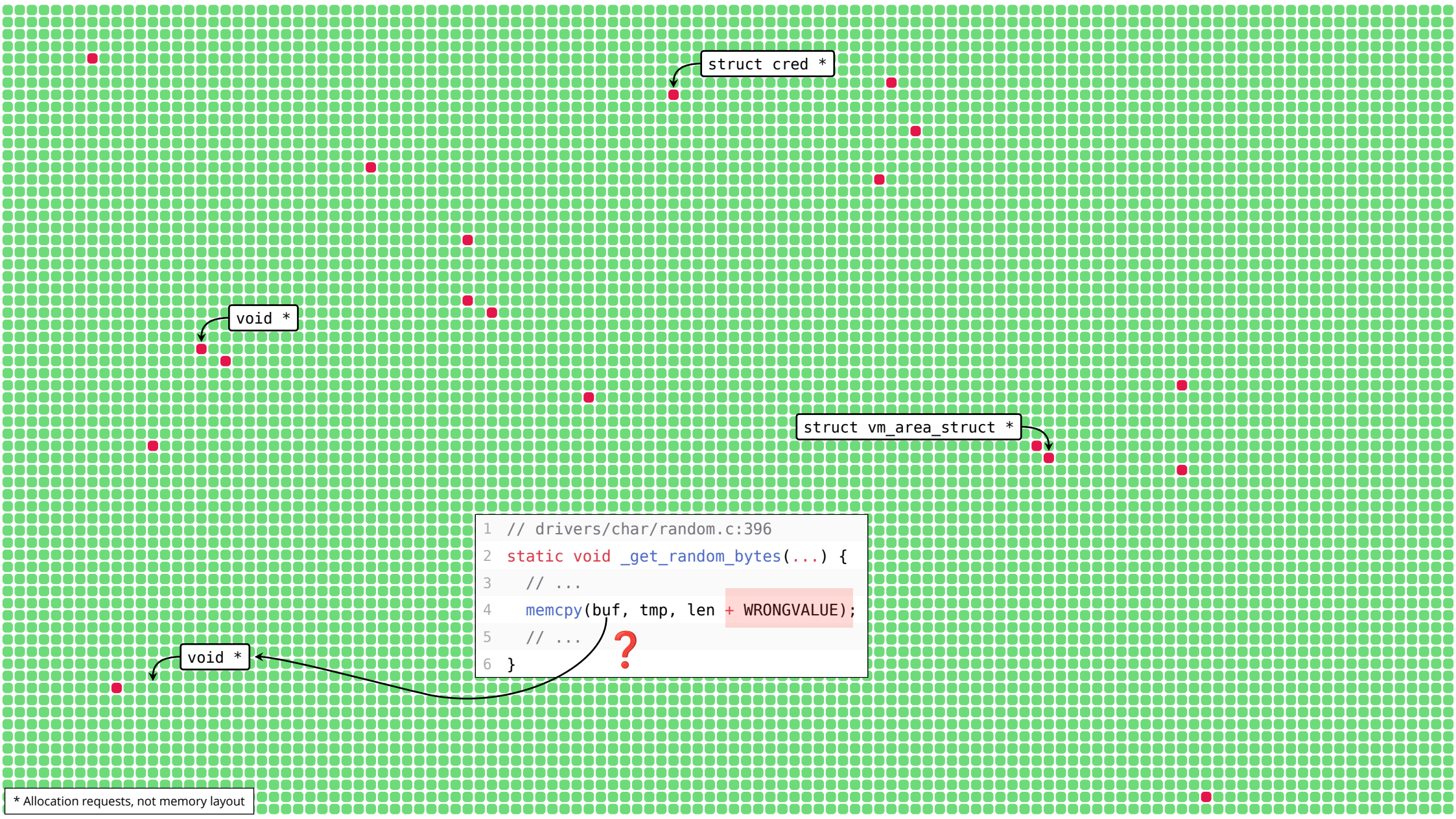

* Allocation requests, not memory layout


struct cred *
void *
struct vm_area_struct *
void *
* Allocation requests, not memory layout

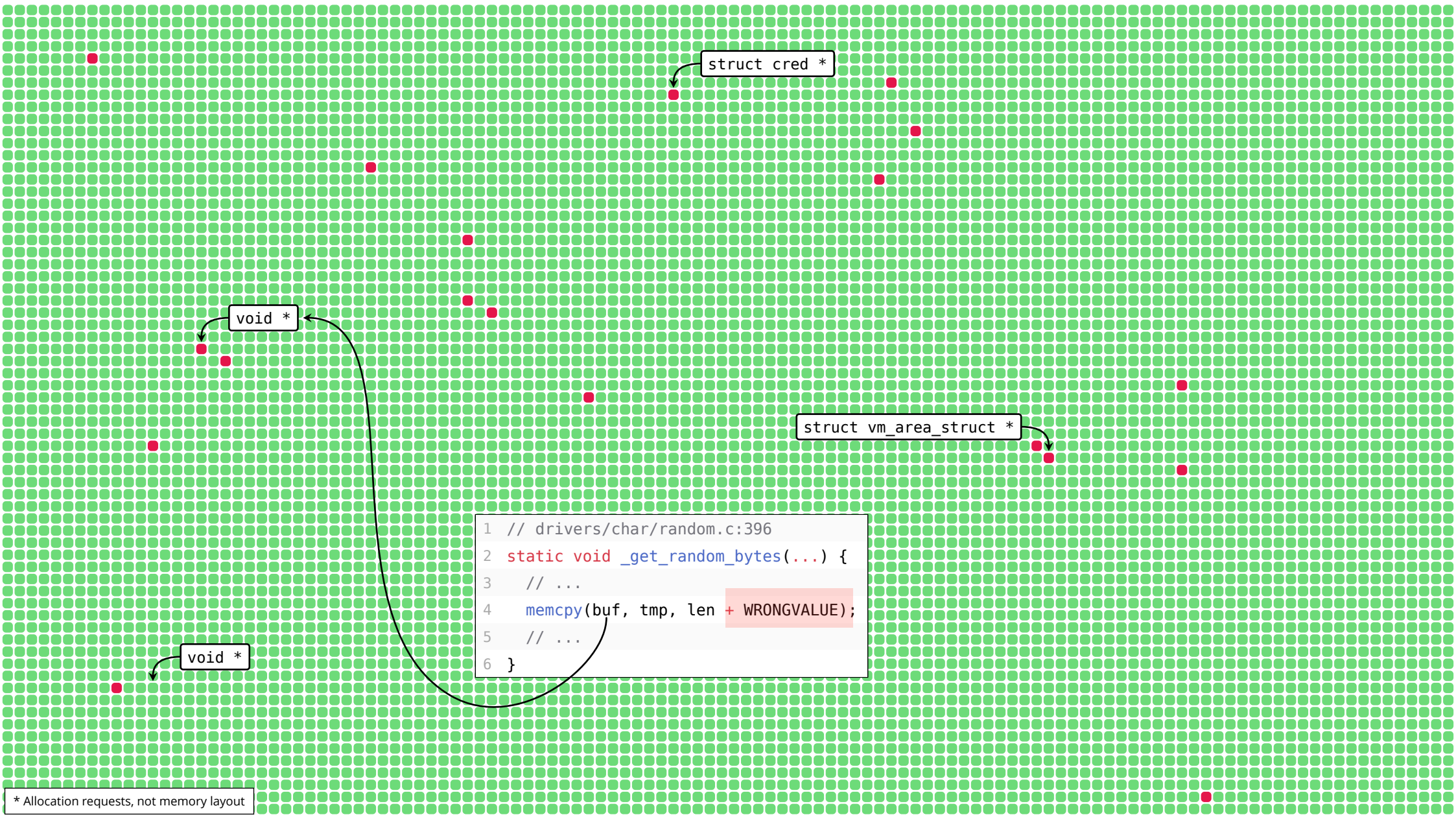

struct cred *
1 // kernel/cred.c:676
2 int set_create_files_as(...) {
3 // ...
4 new->fsuid = inode->i_uid;
5 // ...
6 }
void *
struct vm_area_struct *
void *
* Allocation requests, not memory layout

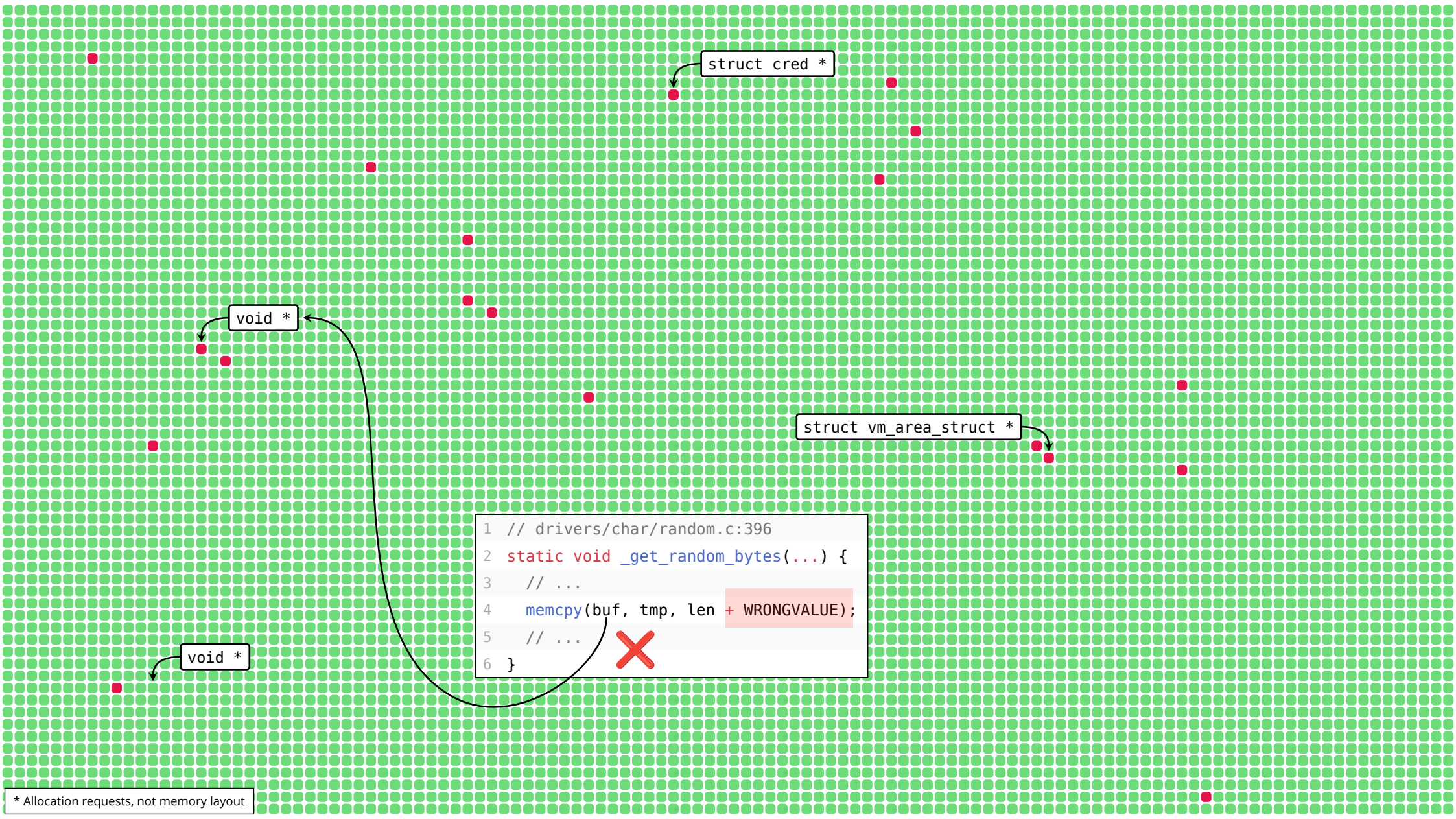

struct cred *
1 // kernel/cred.c:676
2 int set_create_files_as(...) {
3 // ...
4 new->fsuid = inode->i_uid;
5 // ...
6 }
void *
struct vm_area_struct *
void *
* Allocation requests, not memory layout

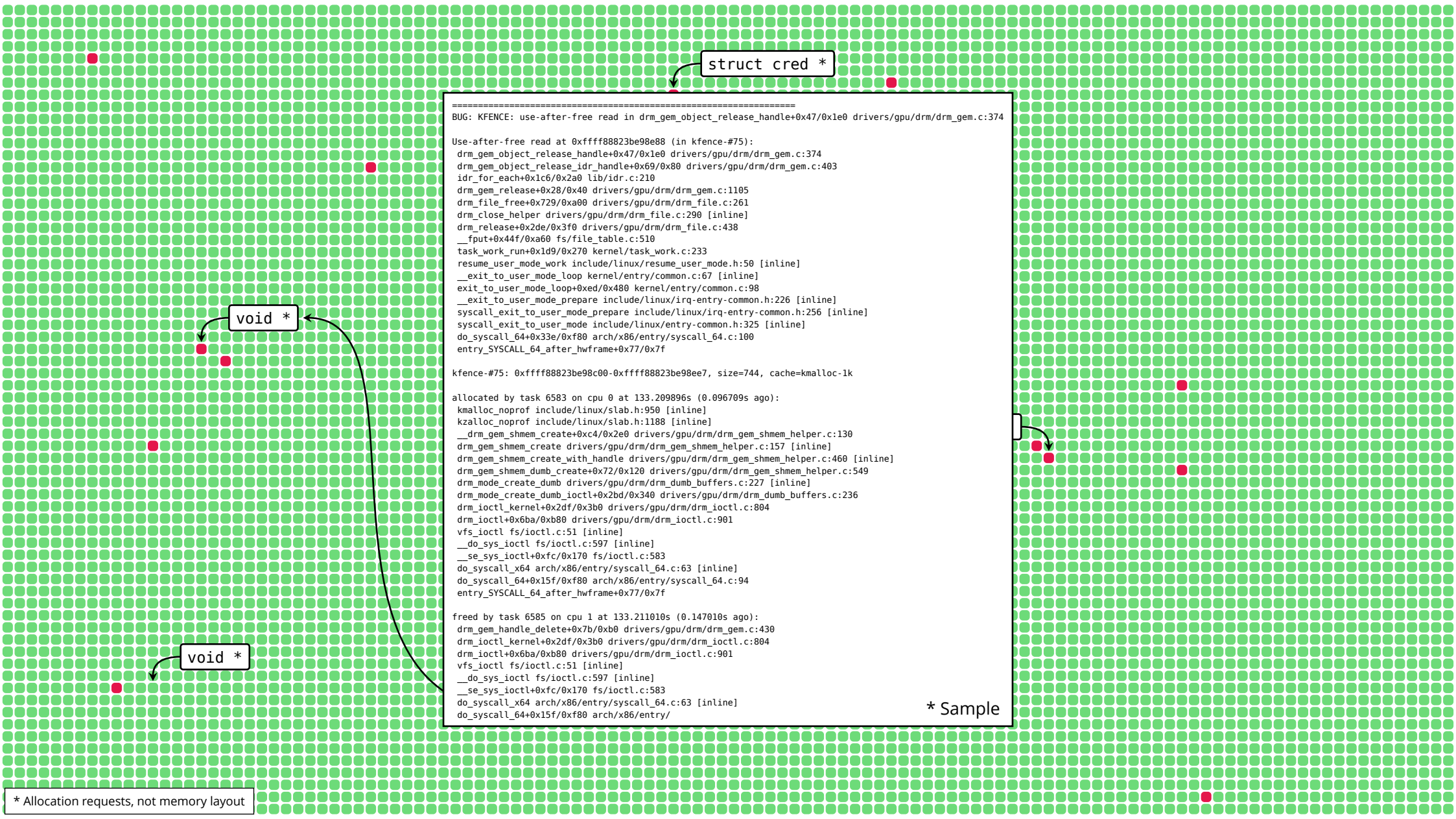

struct cred *
void *
struct vm_area_struct *
1 // drivers/char/random.c:396
2 static void _get_random_bytes(...) {
3 // ...
4 memcpy(buf, tmp, len + WRONGVALUE);
5 // ...
6 }
void *
* Allocation requests, not memory layout

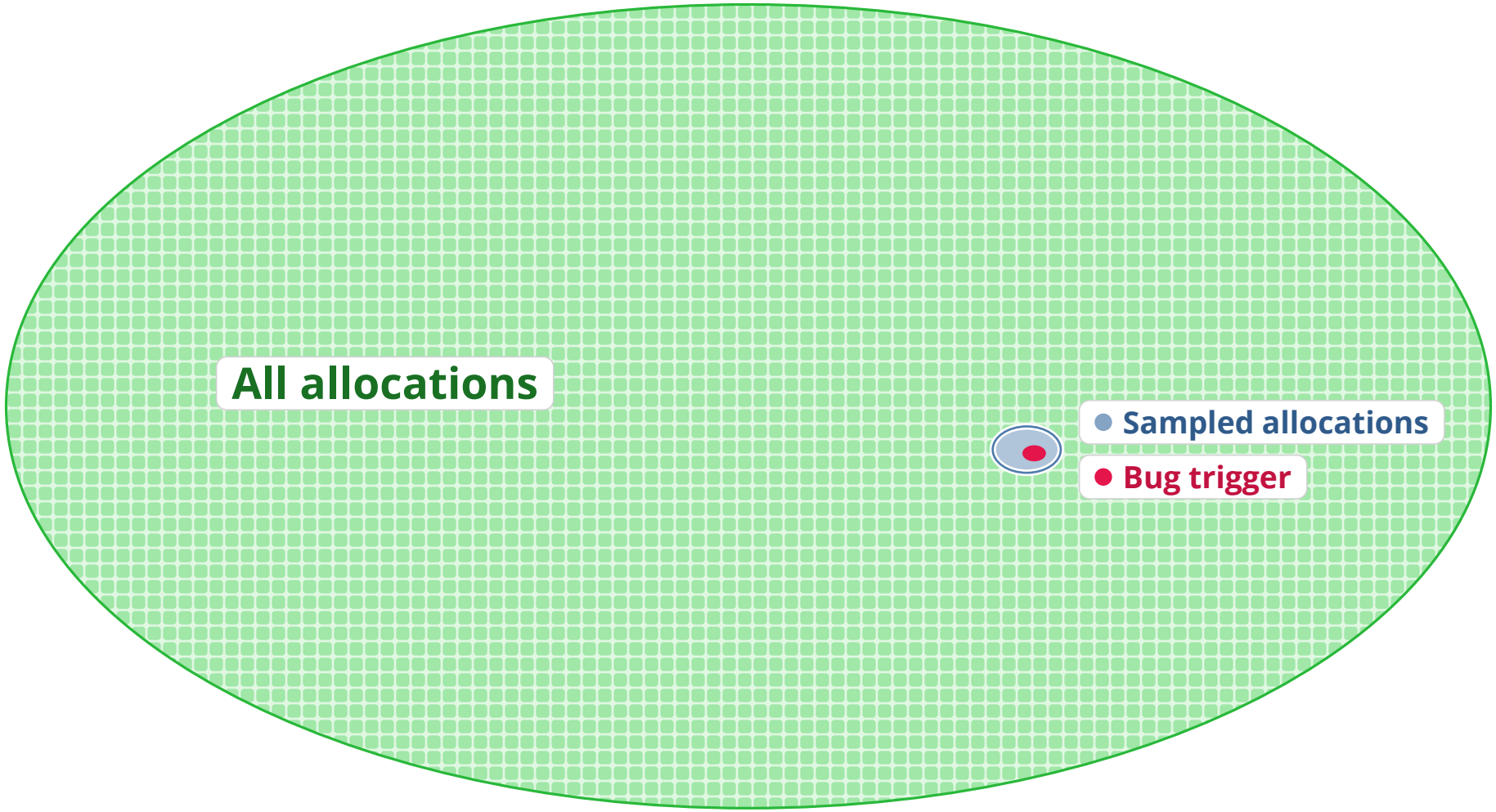


* Allocation requests, not memory layout

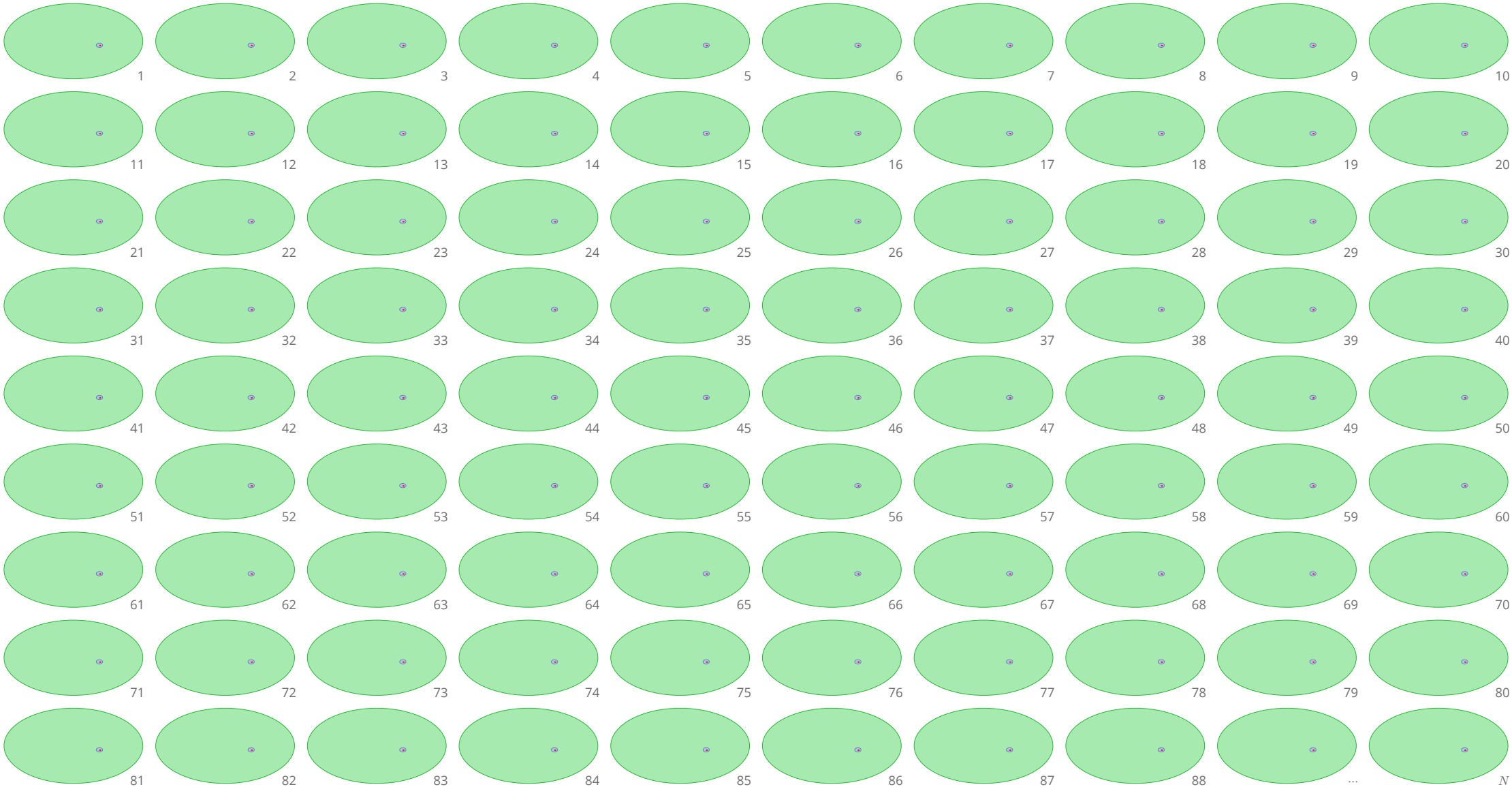

struct cred *
void *
struct vm_area_struct *
void *
1 // drivers/char/random.c:396
2 static void _get_random_bytes(...) {
3 // ...
4 memcpy(buf, tmp, len + WRONGVALUE);
5 // ...
6 }
* Allocation requests, not memory layout

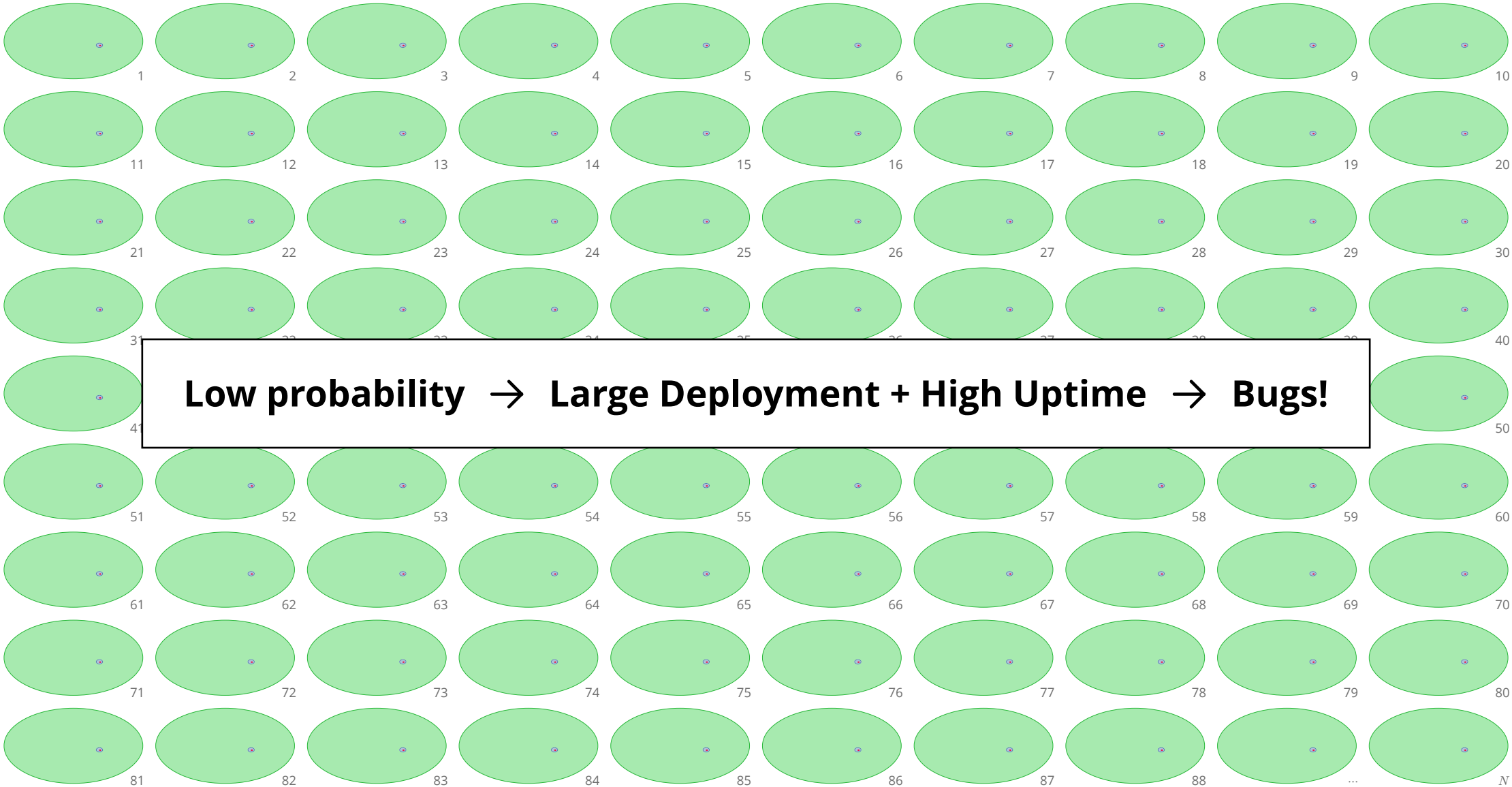

struct cred *
void *
struct vm_area_struct *
1 // drivers/char/random.c:396
2 static void _get_random_bytes(...) {
3 // ...
4 memcpy(buf, tmp, len + WRONGVALUE);
5 // ...
6 }
void *
* Allocation requests, not memory layout


struct cred *
==
BUG: KFENCE: use-after-free read in drm_gem_object_release_handle+0x47/0x1e0 drivers/gpu/drm/drm_gem.c:374
Use-after-free read at 0xffff88823be98e88 (in kfence-#75):
drm_gem_object_release_handle+0x47/0x1e0 drivers/gpu/drm/drm_gem.c:374
drm_gem_object_release_idr_handle+0x69/0x80 drivers/gpu/drm/drm_gem.c:403
idr_for_each+0x1c6/0x2a0 lib/idr.c:210
drm_gem_release+0x28/0x40 drivers/gpu/drm/drm_gem.c:1105
drm_file_free+0x729/0xa00 drivers/gpu/drm/drm_file.c:261
drm_close_helper drivers/gpu/drm/drm_file.c:290 [inline]
drm_release+0x2de/0x3f0 drivers/gpu/drm/drm_file.c:438
__fput+0x44f/0xa60 fs/file_table.c:510
task_work_run+0x1d9/0x270 kernel/task_work.c:233
resume_user_mode_work include/linux/resume_user_mode.h:50 [inline]
__exit_to_user_mode_loop kernel/entry/common.c:67 [inline]
exit_to_user_mode_loop+0xed/0x480 kernel/entry/common.c:98
__exit_to_user_mode_prepare include/linux/irq-entry-common.h:226 [inline]
syscall_exit_to_user_mode_prepare include/linux/irq-entry-common.h:256 [inline]
syscall_exit_to_user_mode include/linux/entry-common.h:325 [inline]
do_syscall_64+0x33e/0xf80 arch/x86/entry/syscall_64.c:100
entry_SYSCALL_64_after_hwframe+0x77/0x7f
kfence-#75: 0xffff88823be98c00-0xffff88823be98ee7, size=744, cache=kmalloc-1k
allocated by task 6583 on cpu 0 at 133.209896s (0.096709s ago):
kmalloc_noprof include/linux/slab.h:950 [inline]
kzalloc_noprof include/linux/slab.h:1188 [inline]
__drm_gem_shmem_create+0xc4/0x2e0 drivers/gpu/drm/drm_gem_shmem_helper.c:130
drm_gem_shmem_create drivers/gpu/drm/drm_gem_shmem_helper.c:157 [inline]
drm_gem_shmem_create_with_handle drivers/gpu/drm/drm_gem_shmem_helper.c:460 [inline]
drm_gem_shmem_dumb_create+0x72/0x120 drivers/gpu/drm/drm_gem_shmem_helper.c:549
drm_mode_create_dumb drivers/gpu/drm/drm_dumb_buffers.c:227 [inline]
drm_mode_create_dumb_ioctl+0x2bd/0x340 drivers/gpu/drm/drm_dumb_buffers.c:236
drm_ioctl_kernel+0x2df/0x3b0 drivers/gpu/drm/drm_ioctl.c:804
drm_ioctl+0x6ba/0xb80 drivers/gpu/drm/drm_ioctl.c:901
vfs_ioctl fs/ioctl.c:51 [inline]
__do_sys_ioctl fs/ioctl.c:597 [inline]
__se_sys_ioctl+0xfc/0x170 fs/ioctl.c:583
do_syscall_x64 arch/x86/entry/syscall_64.c:63 [inline]
do_syscall_64+0x15f/0xf80 arch/x86/entry/syscall_64.c:94
entry_SYSCALL_64_after_hwframe+0x77/0x7f
freed by task 6585 on cpu 1 at 133.211010s (0.147010s ago):
drm_gem_handle_delete+0x7b/0xb0 drivers/gpu/drm/drm_gem.c:430
drm_ioctl_kernel+0x2df/0x3b0 drivers/gpu/drm/drm_ioctl.c:804
drm_ioctl+0x6ba/0xb80 drivers/gpu/drm/drm_ioctl.c:901
vfs_ioctl fs/ioctl.c:51 [inline]
__do_sys_ioctl fs/ioctl.c:597 [inline]
__se_sys_ioctl+0xfc/0x170 fs/ioctl.c:583
do_syscall_x64 arch/x86/entry/syscall_64.c:63 [inline]
do_syscall_64+0x15f/0xf80 arch/x86/entry/
* Sample
void *
void *
* Allocation requests, not memory layout



- Most allocations are fast
- Low probability of detection → sampled AND bug trigger


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
...
N



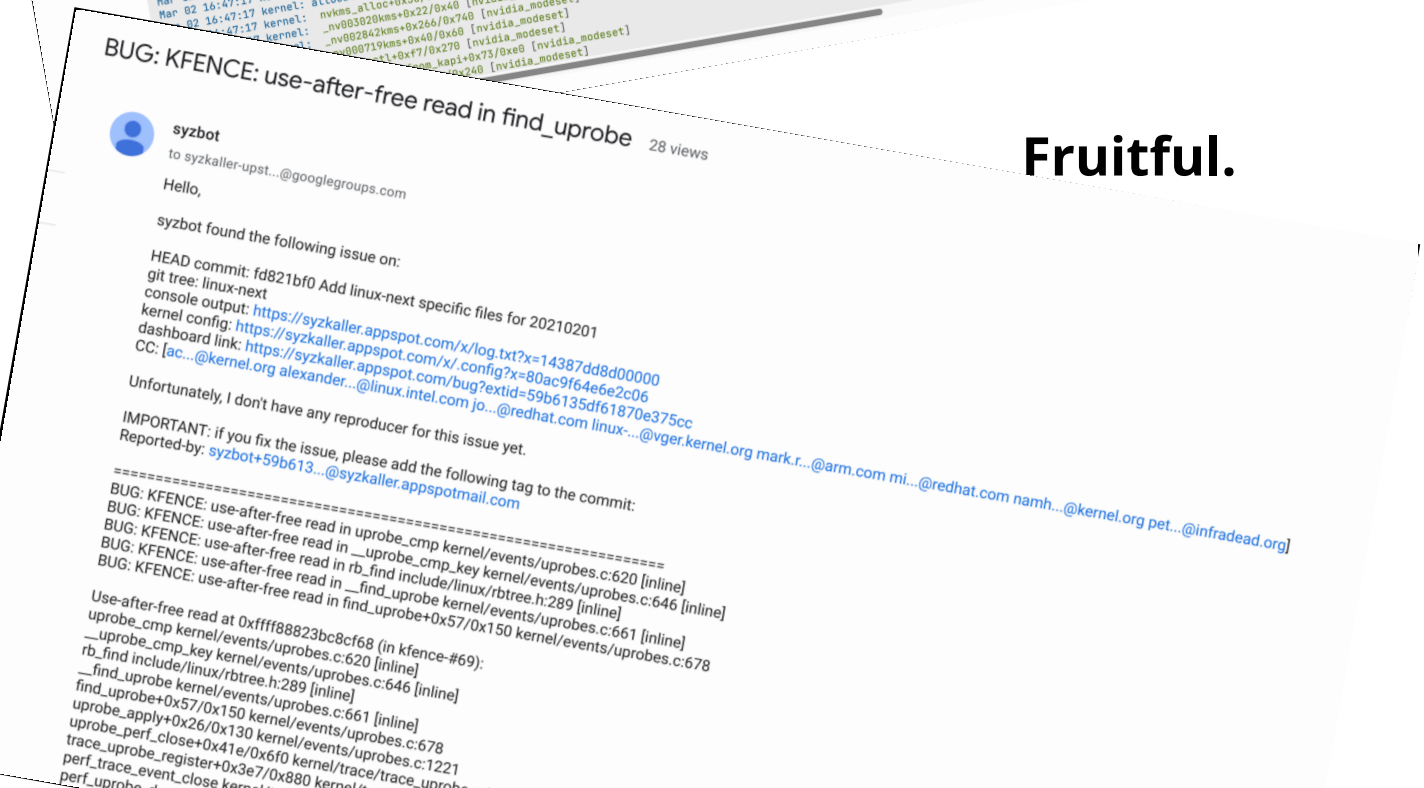
Low probability → Large Deployment + High Uptime → Bugs!

Fruitful.

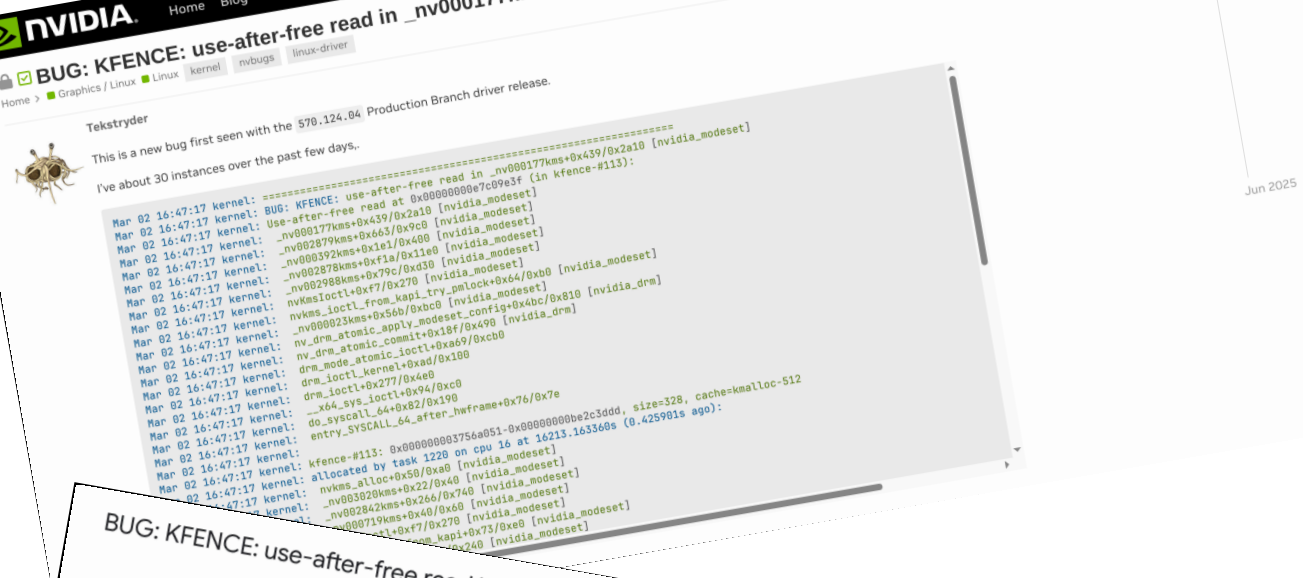

NVIDIA
Home
BUG: KFENCE: use-after-free read in _nv000177...
kernel
nvbugs
linux-driver
Home
Graphics / Linux
Linux
Tekstryder
This is a new bug first seen with the 570.124.04 Production Branch driver release.
I've about 30 instances over the past few days,.
Mar 02 16:47:17 kernel: ==========
Mar 02 16:47:17 kernel: BUG: KFENCE: use-after-free read in _nv000177kms+0x439/0x2a10 [nvidia_modeset]
Mar 02 16:47:17 kernel: Use-after-free read at 0x00000000e7c09e3f (in kfence-#113):
Mar 02 16:47:17 kernel: _nv000177kms+0x439/0x2a10 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv002879kms+0x663/0x9c0 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv000392kms+0x1e1/0x400 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv002878kms+0xf1a/0x11e0 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv002988kms+0x79c/0xd30 [nvidia_modeset]
Mar 02 16:47:17 kernel: nvKmsIoctl+0xf7/0x270 [nvidia_modeset]
Mar 02 16:47:17 kernel: nvkms_ioctl_from_kapi_try_pmlock+0x64/0xb0 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv000023kms+0x56b/0xbc0 [nvidia_modeset]
Mar 02 16:47:17 kernel: nv_drm_atomic_apply_modeset_config+0x4bc/0x810 [nvidia_drm]
Mar 02 16:47:17 kernel: nv_drm_atomic_commit+0x18f/0x490 [nvidia_drm]
Mar 02 16:47:17 kernel: drm_mode_atomic_ioctl+0xad/0x180
Mar 02 16:47:17 kernel: drm_ioctl_kernel+0xad/0x180
Mar 02 16:47:17 kernel: drm_ioctl+0x277/0x4e0
Mar 02 16:47:17 kernel: __x64_sys_ioctl+0x94/0xc0
Mar 02 16:47:17 kernel: do_syscall_64+0x82/0x190
Mar 02 16:47:17 kernel: entry_SYSCALL_64_after_hwframe+0x76/0x7e
Mar 02 16:47:17 kernel:
Mar 02 16:47:17 kernel: kfence-#113: 0x000000003756a051-0x00000000be2c3ddd, size=328, cache=kmalloc-512
Mar 02 16:47:17 kernel: allocated by task 1220 on cpu 16 at 16213.163360s (0.425901s ago):
Mar 02 16:47:17 kernel: nvkms_alloc+0x50/0xa0 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv003020kms+0x22/0x40 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv002842kms+0x266/0x740 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv000719kms+0x40/0x60 [nvidia_modeset]
Mar 02 16:47:17 kernel: nvKmsIoctl+0xf7/0x270 [nvidia_modeset]
Mar 02 16:47:17 kernel: nvkms_ioctl_from_kapi+0x73/0xe0 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv000096kms+0x19d/0x240 [nvidia_modeset]
Jun 2025

Fruitful.

NVIDIA
Home
BUG: KFENCE: use-after-free read in _nv000177
kernel
nvbugs
linux-driver
Home › Graphics / Linux › Linux
Tekstryder
This is a new bug first seen with the 570.124.04 Production Branch driver release.
I've about 30 instances over the past few days,.
Jun 2025
Mar 02 16:47:17 kernel: BUG: KFENCE: use-after-free read at 0x00000000e7c09e3f (in kfence-#113):
Mar 02 16:47:17 kernel: Use-after-free read at 0x00000000e7c09e3f (in kfence-#113):
Mar 02 16:47:17 kernel: _nv000177kms+0x439/0x2a10 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv002879kms+0x663/0x9c0 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv000392kms+0x1e1/0x400 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv002878kms+0xf1a/0x11e0 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv002988kms+0x79c/0x430 [nvidia_modeset]
Mar 02 16:47:17 kernel: nvKmsIoctl+0xf7/0x270 [nvidia_modeset]
Mar 02 16:47:17 kernel: nvKmsIoctl_from_kapi_try_pmlock+0x64/0xb0 [nvidia_modeset]
Mar 02 16:47:17 kernel: nvkms_ioctl_from_kapi+0xbc0 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv000023kms+0x56b/0xbc0 [nvidia_drm]
Mar 02 16:47:17 kernel: nv_drm_atomic_apply_modeset_config+0x4bc/0x810 [nvidia_drm]
Mar 02 16:47:17 kernel: nv_drm_atomic_commit+0x18f/0x490 [nvidia_drm]
Mar 02 16:47:17 kernel: nv_drm_atomic_ioctl+0xa69/0xcb0
Mar 02 16:47:17 kernel: drm_mode_atomic_ioctl+0xad/0x100
Mar 02 16:47:17 kernel: drm_ioctl_kernel+0xad/0x180
Mar 02 16:47:17 kernel: drm_ioctl+0x277/0x4e0
Mar 02 16:47:17 kernel: drm_ioctl+0x94/0xc0
Mar 02 16:47:17 kernel: __x64_sys_ioctl+0x190
Mar 02 16:47:17 kernel: do_syscall_64+0x82/0x190
Mar 02 16:47:17 kernel: entry_SYSCALL_64_after_hwframe+0x76/0x7e
Mar 02 16:47:17 kernel:
Mar 02 16:47:17 kernel: kfence-#113: 0x000000003756a051-0x00000000be2c3ddd, size=328, cache=kmalloc-512
Mar 02 16:47:17 kernel: allocated by task 1220 on cpu 16 at 16213.163360s (0.425901s ago):
Mar 02 16:47:17 kernel: nvkms_alloc+0x50/0xa0 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv003020kms+0x22/0x40 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv002842kms+0x266/0x740 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv000719kms+0x40/0x60 [nvidia_modeset]
Mar 02 16:47:17 kernel: nvKmsIoctl+0xf7/0x270 [nvidia_modeset]
Mar 02 16:47:17 kernel: nvKmsIoctl_from_kapi+0x73/0xe0 [nvidia_modeset]
Mar 02 16:47:17 kernel: nvkms_ioctl_from_kapi+0x40 [nvidia_modeset]


BUG: KFENCE: use-after-free read in find_uprobe
28 views
syzbot
to syzkaller-upst...@googlegroups.com
Hello,
syzbot found the following issue on:
HEAD commit: fd821bf0 Add linux-next specific files for 20210201
git tree: linux-next
console output: https://syzkaller.appspot.com/x/log.txt?x=14387dd8d00000
kernel config: https://syzkaller.appspot.com/x/.config?x=80ac9f64e6e2c06
dashboard link: https://syzkaller.appspot.com/bug?extid=59b6135df61870e375cc
CC: [ac...@kernel.org alexander...@linux.intel.com jo...@redhat.com linux-...@vger.kernel.org mark.r...@arm.com mi...@redhat.com namh...@kernel.org pet...@infradead.org]
Unfortunately, I don't have any reproducer for this issue yet.
IMPORTANT: if you fix the issue, please add the following tag to the commit:
Reported-by: syzbot+59b613...@syzkaller.appspotmail.com
==========
BUG: KFENCE: use-after-free read in uprobe_cmp kernel/events/uprobes.c:620 [inline]
BUG: KFENCE: use-after-free read in __uprobe_cmp_key kernel/events/uprobes.c:646 [inline]
BUG: KFENCE: use-after-free read in rb_find include/linux/rbtree.h:289 [inline]
BUG: KFENCE: use-after-free read in __find_uprobe kernel/events/uprobes.c:661 [inline]
BUG: KFENCE: use-after-free read in find_uprobe+0x57/0x150 kernel/events/uprobes.c:678
Use-after-free read at 0xffff88823bc8cf68 (in kfence-#69):
uprobe_cmp kernel/events/uprobes.c:620 [inline]
__uprobe_cmp_key kernel/events/uprobes.c:646 [inline]
rb_find include/linux/rbtree.h:289 [inline]
__find_uprobe kernel/events/uprobes.c:661 [inline]
find_uprobe+0x57/0x150 kernel/events/uprobes.c:678
uprobe_apply+0x26/0x130 kernel/events/uprobes.c:1221
uprobe_perf_close+0x41e/0x6f0 kernel/trace/trace_uprobe.c:1
trace_uprobe_register+0x3e7/0x880 kernel/trace/trace_uprobe.c:1
perf_trace_event_close kernel/trace/trace_uprobe.c:1

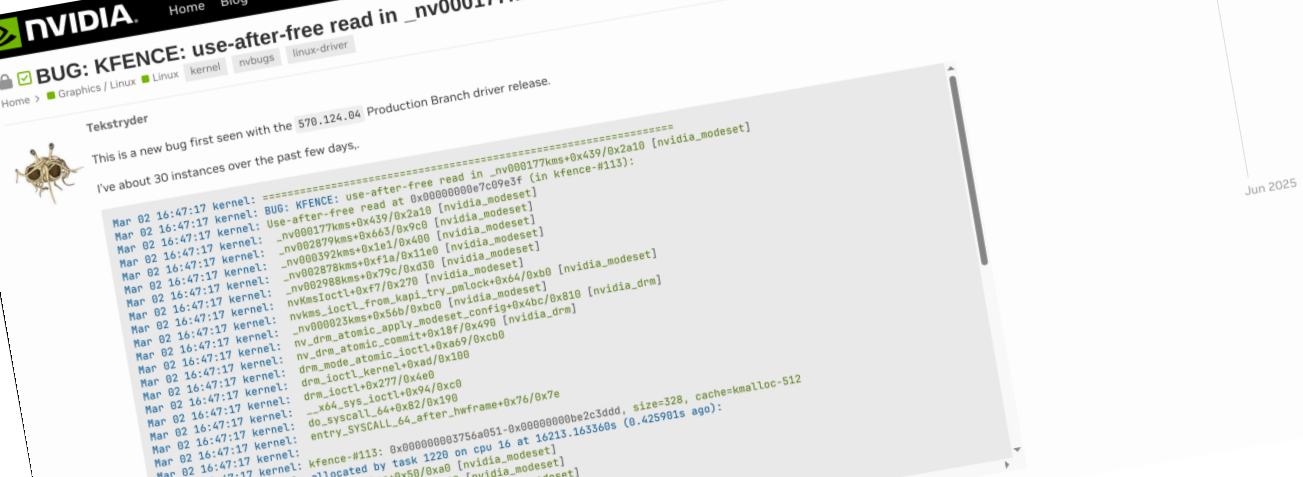
Fruitful.

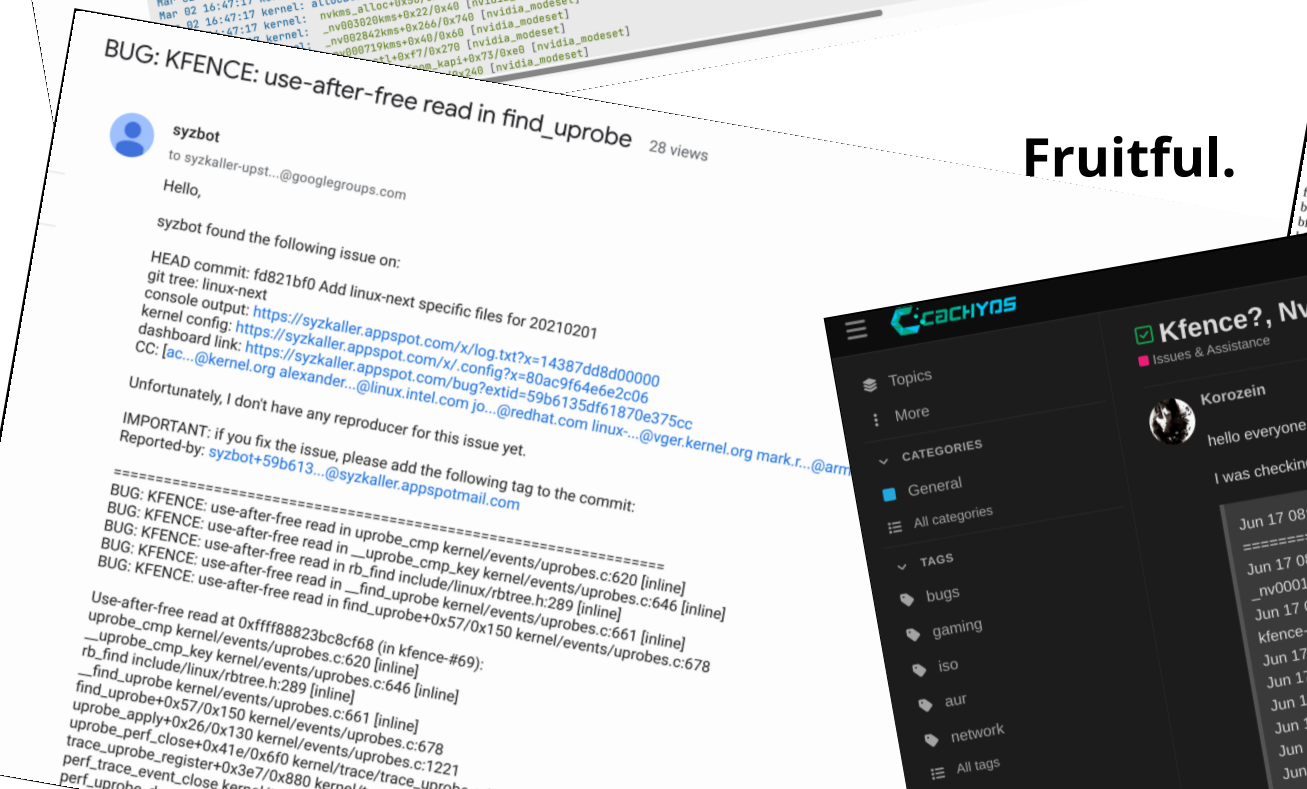

NVIDIA
Home
BUG: KFENCE: use-after-free read in _nv000177
kernel
nvbugs
linux-driver
Home
Graphics / Linux
Linux
Tekstryder
This is a new bug first seen with the 570.124.04 Production Branch driver release.
I've about 30 instances over the past few days,.
Jun 2025
Mar 02 16:47:17 kernel: BUG: KFENCE: use-after-free read in _nv000177kms+0x439/0x2a10 [nvidia_modeset]
Mar 02 16:47:17 kernel: Use-after-free read at 0x00000000e7c09e3f (in kfence-#113):
Mar 02 16:47:17 kernel: _nv000177kms+0x439/0x2a10 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv002879kms+0x1e1/0x400 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv000392kms+0xf1a/0x11e0 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv002878kms+0x79c/0xd30 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv002988kms+0x7f7/0x270 [nvidia_modeset]
Mar 02 16:47:17 kernel: nvKmsIoctl+0xf7/0x270 [nvidia_modeset]
Mar 02 16:47:17 kernel: nvkms_ioctl_from_kapi_try_pmlock+0x64/0xb0 [nvidia_modeset]
Mar 02 16:47:17 kernel: nvkms_ioctl+0x56b/0xbc0 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv000023kms+0x56b/0xbc0 [nvidia_modeset]
Mar 02 16:47:17 kernel: nv_drm_atomic_apply_modeset_config+0x4bc/0x810 [nvidia_drm]
Mar 02 16:47:17 kernel: nv_drm_atomic_commit+0x18f/0x490 [nvidia_drm]
Mar 02 16:47:17 kernel: drm_mode_atomic_ioctl+0xa69/0xcb0
Mar 02 16:47:17 kernel: drm_atomic_ioctl_kernel+0xad/0x180
Mar 02 16:47:17 kernel: drm_ioctl_kernel+0xad/0x100
Mar 02 16:47:17 kernel: drm_ioctl+0x277/0x4e0
Mar 02 16:47:17 kernel: __x64_sys_ioctl+0x94/0xc0
Mar 02 16:47:17 kernel: do_syscall_64+0x82/0x190
Mar 02 16:47:17 kernel: entry_SYSCALL_64_after_hwframe+0x76/0x7e
Mar 02 16:47:17 kernel: kfence-#113: 0x0000000003756a051-0x00000000be2c3ddd, size=328, cache=kmalloc-512
Mar 02 16:47:17 kernel: allocated by task 1220 on cpu 16 at 16213.163360s (0.425901s ago):
Mar 02 16:47:17 kernel: nvkms_alloc+0x50/0xa0 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv003020kms+0x22/0x40 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv002842kms+0x266/0x740 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv000719kms+0x40/0x60 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv000719kms+0xf7/0x270 [nvidia_modeset]
Mar 02 16:47:17 kernel: nvKmsIoctl+0xf7/0x270 [nvidia_modeset]
Mar 02 16:47:17 kernel: nvkms_ioctl_from_kapi+0x73/0xe0 [nvidia_modeset]
Mar 02 16:47:17 kernel: _nv000023kms+0x73/0xe0 [nvidia_modeset]


BUG: KFENCE: use-after-free read in find_uprobe
28 views
syzbot
to syzkaller-upst...@googlegroups.com
Hello,
syzbot found the following issue on:
HEAD commit: fd821bf0 Add linux-next specific files for 20210201
git tree: linux-next
console output: https://syzkaller.appspot.com/x/log.txt?x=14387dd8d00000
kernel config: https://syzkaller.appspot.com/x/.config?x=80ac9f64e6e2c06
dashboard link: https://syzkaller.appspot.com/bug?extid=59b6135df61870e375cc
CC: [ac...@kernel.org alexander...@linux.intel.com jo...@redhat.com linux-...@vger.kernel.org mark.r...@arm.com mi...@redhat.com namh...@kernel.org pet...@infradead.org]
Unfortunately, I don't have any reproducer for this issue yet.
IMPORTANT: if you fix the issue, please add the following tag to the commit:
Reported-by: syzbot+59b613...@syzkaller.appspotmail.com
==========
BUG: KFENCE: use-after-free read in uprobe_cmp kernel/events/uprobes.c:620 [inline]
BUG: KFENCE: use-after-free read in __uprobe_cmp_key kernel/events/uprobes.c:646 [inline]
BUG: KFENCE: use-after-free read in rb_find include/linux/rbtree.h:289 [inline]
BUG: KFENCE: use-after-free read in __find_uprobe kernel/events/uprobes.c:661 [inline]
BUG: KFENCE: use-after-free read in find_uprobe+0x57/0x150 kernel/events/uprobes.c:678
Use-after-free read at 0xffff88823bc8cf68 (in kfence-#69):
uprobe_cmp kernel/events/uprobes.c:620 [inline]
__uprobe_cmp_key kernel/events/uprobes.c:646 [inline]
rb_find include/linux/rbtree.h:289 [inline]
__find_uprobe kernel/events/uprobes.c:661 [inline]
find_uprobe+0x57/0x150 kernel/events/uprobes.c:678
uprobe_apply+0x26/0x130 kernel/events/uprobes.c:1221
uprobe_perf_close+0x41e/0x6f0 kernel/trace/trace_uprobe.c:1
trace_uprobe_register+0x3e7/0x880 kernel/trace/trace_uprobe.c:1
perf_trace_event_close kernel/trace/trace_uprobe.c:1

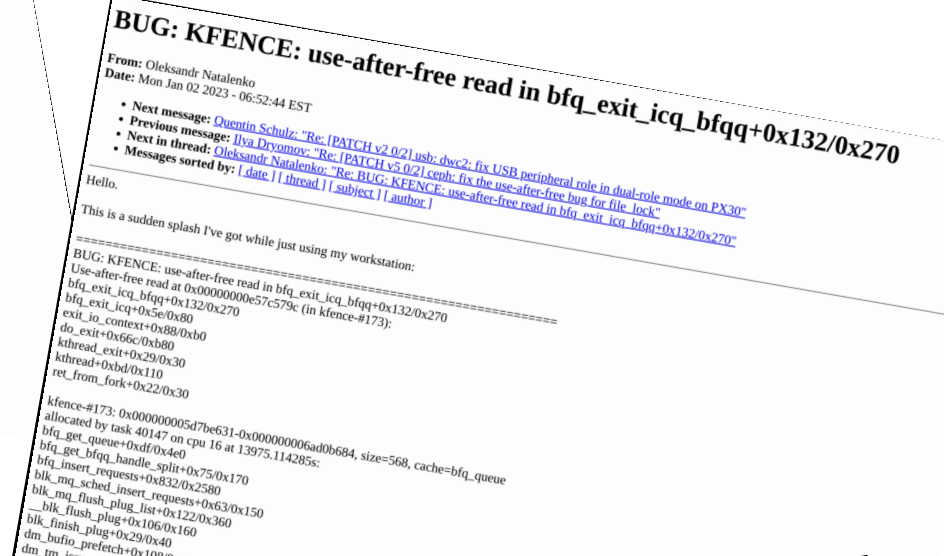
Fruitful.

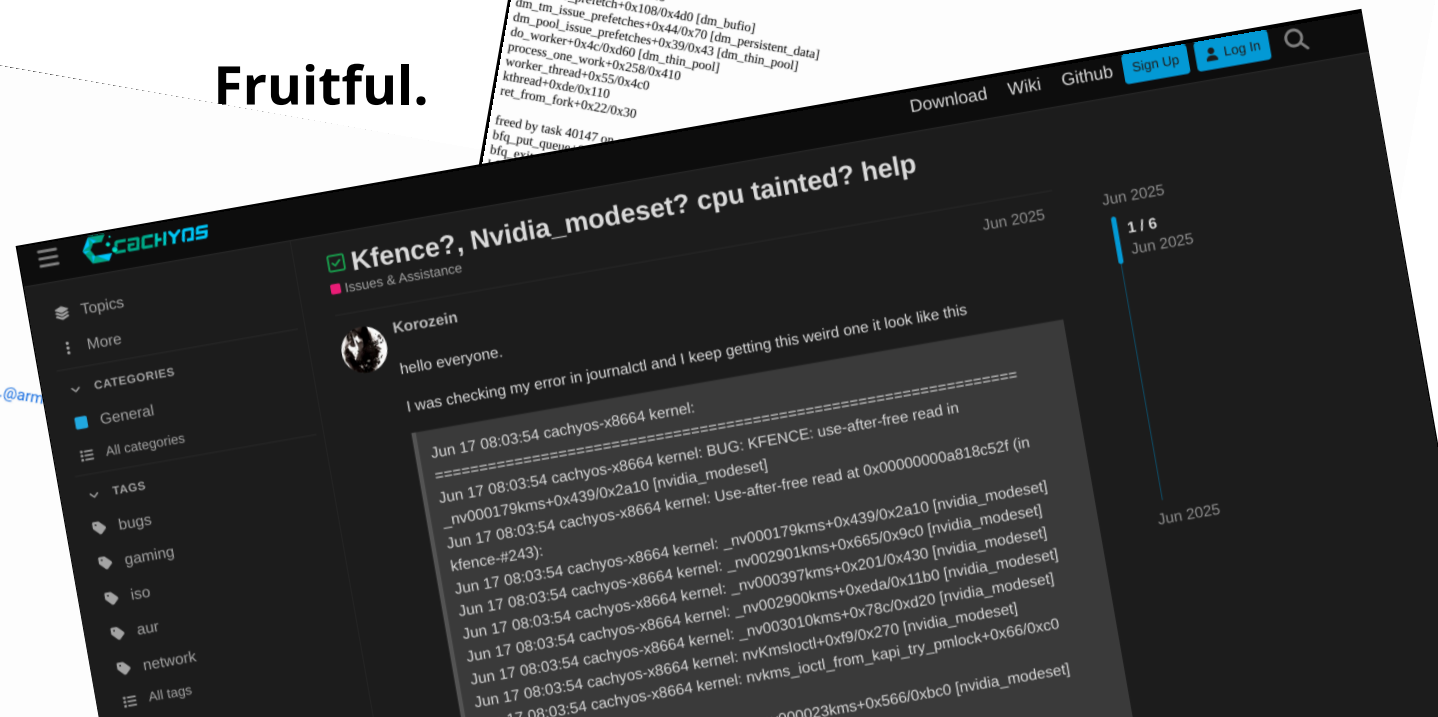

BUG: KFENCE: use-after-free read in bfq_exit_icq_bfqq+0x132/0x270
From: Oleksandr Natalenko
Date: Mon Jan 02 2023 - 06:52:44 EST
Next message: Quentin Schulz: "Re: [PATCH v2 0/2] usb: dwc2: fix USB peripheral role in dual-role mode on PX30"
Previous message: Ilya Dryomov: "Re: [PATCH v5 0/2] ceph: fix the use-after-free bug for file_lock"
Next in thread: Oleksandr Natalenko: "Re: BUG: KFENCE: use-after-free read in bfq_exit_icq_bfqq+0x132/0x270"
Messages sorted by: [date] [thread] [subject] [author]
Hello.
This is a sudden splash I've got while just using my workstation:
==========
BUG: KFENCE: use-after-free read in bfq_exit_icq_bfqq+0x132/0x270
Use-after-free read at 0x00000000e57c579c (in kfence-#173):
bfq_exit_icq_bfqq+0x132/0x270
bfq_exit_icq+0x5e/0x80
exit_io_context+0x88/0xb0
do_exit+0x66c/0xb80
kthread_exit+0x29/0x30
kthread+0xbd/0x110
ret_from_fork+0x22/0x30
kfence-#173: 0x0000000005d7be631-0x000000006ad0b684, size=568, cache=bfq_queue
allocated by task 40147 on cpu 16 at 13975.114285s:
bfq_get_queue+0xdf/0x4e0
bfq_get_bfqq_handle_split+0x75/0x170
bfq_insert_requests+0x832/0x2580
blk_mq_sched_insert_requests+0x63/0x150
blk_mq_flush_plug_list+0x122/0x360
__blk_flush_plug+0x106/0x160
blk_finish_plug+0x29/0x40
dm_bufio_prefetch+0x108/0x4d0 [dm_bufio]
dm_tm_issue_prefetches+0x44/0x70 [dm_persistent_data]
dm_pool_issue_prefetches+0x39/0x43 [dm_thin_pool]
do_worker+0x4c/0xd60 [dm_thin_pool]
process_one_work+0x258/0x410
worker_thread+0x55/0x4c0
kthread+0xde/0x110
ret_from_fork+0x22/0x30
freed by task 40147 on cpu 20 at 14500.096700s:
bfq_put_queue+0x185/0x2d0
bfq_exit_icq_bfqq+0x129/0x270
bfq_exit_icq+0x5e/0x80
exit_io_context+0x88/0xb0
do_exit+0x66c/0xb80
kthread_exit+0x29/0x30
kthread+0xbd/0x110
ret_from_fork+0x22/0x30

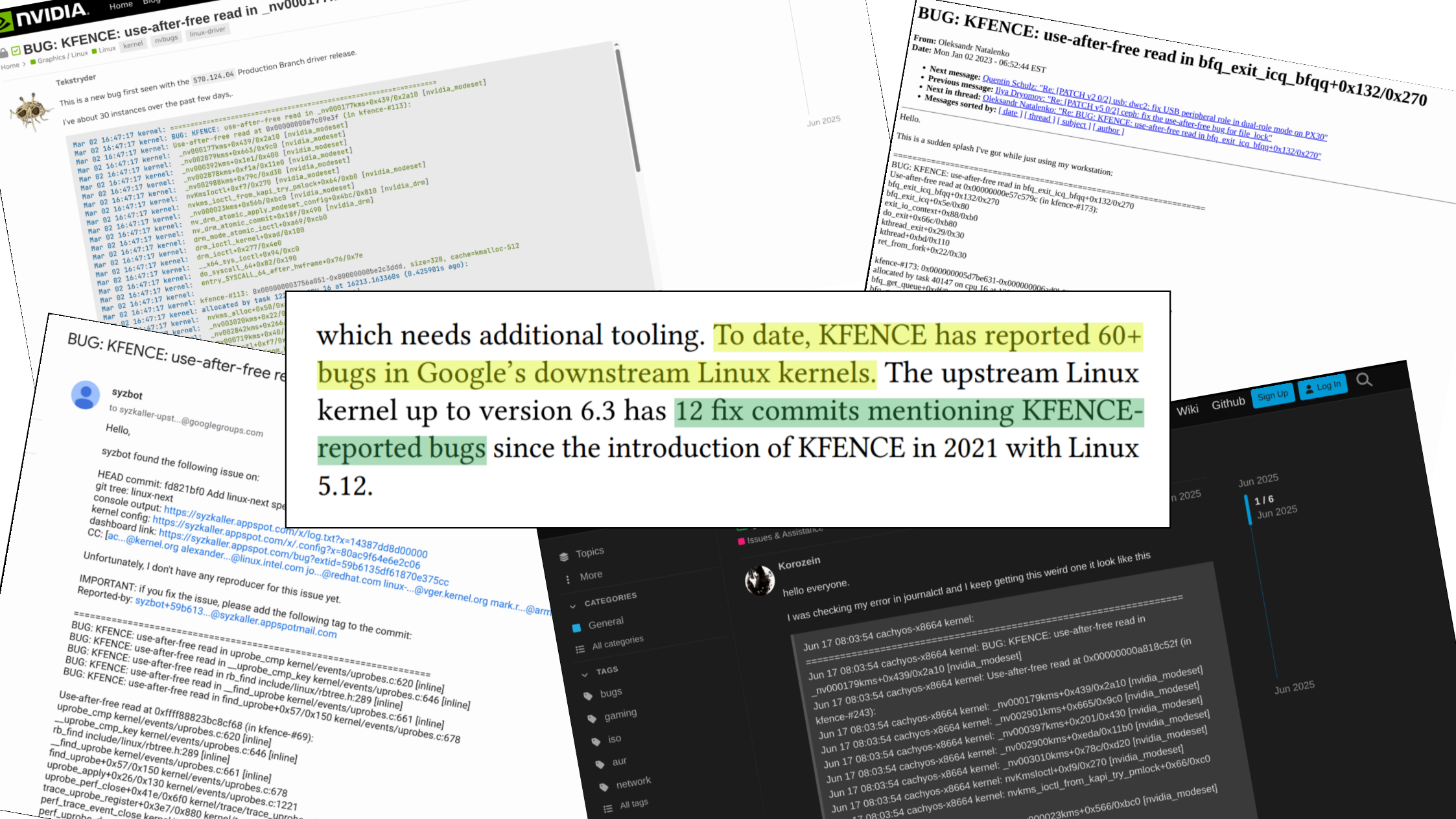

NVIDIA
Home
BUG: KFENCE: use-after-free read in _nv000177
kernel
nvbugs
linux-driver
Home › Graphics / Linux › Linux
Tekstryder
This is a new bug first seen with the 570.124.04 Production Branch driver release.
I've about 30 instances over the past few days,.
Jun 2025


BUG: KFENCE: use-after-free read in find_uprobe
28 views
syzbot
to syzkaller-upst...@googlegroups.com
Hello,
syzbot found the following issue on:
HEAD commit: fd821bf0 Add linux-next specific files for 20210201
git tree: linux-next
console output: https://syzkaller.appspot.com/x/log.txt?x=14387dd8d00000
kernel config: https://syzkaller.appspot.com/x/.config?x=80ac9f64e6e2c06
dashboard link: https://syzkaller.appspot.com/bug?extid=59b6135df61870e375cc
CC: [ac...@kernel.org alexander...@linux.intel.com jo...@redhat.com linux-...@vger.kernel.org mark.r...@arm
Unfortunately, I don't have any reproducer for this issue yet.
IMPORTANT: if you fix the issue, please add the following tag to the commit:
Reported-by: syzbot+59b613...@syzkaller.appspotmail.com
BUG: KFENCE: use-after-free read in uprobe_cmp kernel/events/uprobes.c:620 [inline]
BUG: KFENCE: use-after-free read in __uprobe_cmp_key kernel/events/uprobes.c:646 [inline]
BUG: KFENCE: use-after-free read in rb_find include/linux/rbtree.h:289 [inline]
BUG: KFENCE: use-after-free read in __find_uprobe kernel/events/uprobes.c:661 [inline]
BUG: KFENCE: use-after-free read in find_uprobe+0x57/0x150 kernel/events/uprobes.c:678
Use-after-free read at 0xffff88823bc8cf68 (in kfence-#69):
uprobe_cmp kernel/events/uprobes.c:620 [inline]
__uprobe_cmp_key kernel/events/uprobes.c:646 [inline]
rb_find include/linux/rbtree.h:289 [inline]
__find_uprobe kernel/events/uprobes.c:661 [inline]
find_uprobe+0x57/0x150 kernel/events/uprobes.c:678
uprobe_apply+0x26/0x130 kernel/events/uprobes.c:1221
uprobe_perf_close+0x41e/0x6f0 kernel/trace/trace_uprobe.c
trace_uprobe_register+0x3e7/0x880 kernel/trace/trace_uprobe.c
perf_trace_event_close kernel/trace/trace_event_perf.c

Fruitful.


BUG: KFENCE: use-after-free read in bfq_exit_icq_bfqq+0x132/0x270
From: Oleksandr Natalenko
Date: Mon Jan 02 2023 - 06:52:44 EST
Next message: Quentin Schulz: "Re: [PATCH v2 0/2] usb: dwc2: fix USB peripheral role in dual-role mode on PX30"
Previous message: Ilya Dryomov: "Re: [PATCH v5 0/2] ceph: fix the use-after-free bug for file_lock"
Next in thread: Oleksandr Natalenko: "Re: BUG: KFENCE: use-after-free read in bfq_exit_icq_bfqq+0x132/0x270"
Messages sorted by: [date] [thread] [subject] [author]
Hello.
This is a sudden splash I've got while just using my workstation:
BUG: KFENCE: use-after-free read in bfq_exit_icq_bfqq+0x132/0x270
Use-after-free read at 0x00000000e57c579c (in kfence-#173):
bfq_exit_icq_bfqq+0x132/0x270
bfq_exit_icq+0x5e/0x80
exit_io_context+0x88/0xb0
do_exit+0x66c/0xb80
kthread_exit+0x29/0x30
kthread+0xbd/0x110
ret_from_fork+0x22/0x30
kfence-#173: 0x0000000005d7be631-0x000000006ad0b684, size=568, cache=bfq_queue
allocated by task 40147 on cpu 16 at 13975.114285s:
bfq_get_queue+0xdf/0x4e0
bfq_get_bfqq_handle_split+0x75/0x170
bfq_insert_requests+0x832/0x2580
blk_mq_sched_insert_requests+0x63/0x150
blk_mq_flush_plug_list+0x122/0x360
__blk_flush_plug+0x106/0x160
blk_finish_plug+0x29/0x40
dm_bufio_prefetch+0x108/0x4d0 [dm_bufio]
dm_tm_issue_prefetches+0x44/0x70 [dm_persistent_data]
dm_pool_issue_prefetches+0x39/0x43 [dm_thin_pool]
do_worker+0x4c/0xd60 [dm_thin_pool]
process_one_work+0x258/0x410
worker_thread+0x55/0x4c0
kthread+0xde/0x110
ret_from_fork+0x22/0x30
freed by task 40147 on


cachyos
Download
Wiki
Github
Sign Up
Log In
Topics
More
CATEGORIES
General
All categories
TAGS
bugs
gaming
iso
aur
network
All tags
Kfence?, Nvidia_modeset? cpu tainted? help
Issues & Assistance
Jun 2025
Korozein
hello everyone.
I was checking my error in journalctl and I keep getting this weird one it look like this
Jun 17 08:03:54 cachyos-x8664 kernel:
Jun 17 08:03:54 cachyos-x8664 kernel: BUG: KFENCE: use-after-free read in
_nv000179kms+0x439/0x2a10 [nvidia_modeset]
Jun 17 08:03:54 cachyos-x8664 kernel: Use-after-free read at 0x00000000a818c52f (in
kfence-#243):
Jun 17 08:03:54 cachyos-x8664 kernel: _nv000179kms+0x439/0x2a10 [nvidia_modeset]
Jun 17 08:03:54 cachyos-x8664 kernel: _nv002901kms+0x665/0x9c0 [nvidia_modeset]
Jun 17 08:03:54 cachyos-x8664 kernel: _nv000397kms+0x201/0x430 [nvidia_modeset]
Jun 17 08:03:54 cachyos-x8664 kernel: _nv002900kms+0xeda/0x11b0 [nvidia_modeset]
Jun 17 08:03:54 cachyos-x8664 kernel: _nv003010kms+0x78c/0xd20 [nvidia_modeset]
Jun 17 08:03:54 cachyos-x8664 kernel: nvKmsIoctl+0xf9/0x270 [nvidia_modeset]
Jun 17 08:03:54 cachyos-x8664 kernel: nvkms_ioctl_from_kapi_try_pmlock+0x66/0xc0
Jun 17 08:03:54 cachyos-x8664 kernel: _nv000023kms+0x566/0xbc0 [nvidia_modeset]
1 / 6
Jun 2025
Jun 2025


NVIDIA
Home
BUG: KFENCE: use-after-free read in _nv000177
kernel
nvbugs
linux-driver
Tekstryder
This is a new bug first seen with the 570.124.04 Production Branch driver release.
I've about 30 instances over the past few days,.
Jun 2025
BUG: KFENCE: use-after-free read in bfq_exit_icq_bfqq+0x132/0x270
From: Oleksandr Natalenko
Date: Mon Jan 02 2023 - 06:52:44 EST
Next message: Quentin Schulz: "Re: [PATCH v2 0/2] usb: dwc2: fix USB peripheral role in dual-role mode on PX30"
Previous message: Ilya Dryomov: "Re: [PATCH v5 0/2] ceph: fix the use-after-free bug for file_lock"
Next in thread: Oleksandr Natalenko: "Re: BUG: KFENCE: use-after-free read in bfq_exit_icq_bfqq+0x132/0x270"
Messages sorted by: [date] [thread] [subject] [author]
Hello.
This is a sudden splash I've got while just using my workstation:
BUG: KFENCE: use-after-free read in bfq_exit_icq_bfqq+0x132/0x270
Use-after-free read at 0x00000000e57c579c (in kfence-#173):
bfq_exit_icq_bfqq+0x132/0x270
bfq_exit_icq+0x5e/0x80
exit_io_context+0x88/0xb0
do_exit+0x66c/0xb80
kthread_exit+0x29/0x30
kthread+0xbd/0x110
ret_from_fork+0x22/0x30
kfence-#173: 0x0000000005d7be631-0x00000000f6a4d1
allocated by task 40147 on cpu 16 at
bfq_get_queue+0x
which needs additional tooling. To date, KFENCE has reported 60+ bugs in Google's downstream Linux kernels. The upstream Linux kernel up to version 6.3 has 12 fix commits mentioning KFENCE-reported bugs since the introduction of KFENCE in 2021 with Linux 5.12.
BUG: KFENCE: use-after-free re
syzbot
to syzkaller-upst...@googlegroups.com
Hello,
syzbot found the following issue on:
HEAD commit: fd821bf0 Add linux-next spe
git tree: linux-next
console output: https://syzkaller.appspot.com/x/log.txt?x=14387dd8d00000
kernel config: https://syzkaller.appspot.com/x/.config?x=80ac9f64e6e2c06
dashboard link: https://syzkaller.appspot.com/bug?extid=59b6135df61870e375cc
CC: [ac...@kernel.org alexander...@linux.intel.com jo...@redhat.com linux-...@vger.kernel.org mark.r...@arm
Unfortunately, I don't have any reproducer for this issue yet.
IMPORTANT: if you fix the issue, please add the following tag to the commit:
Reported-by: syzbot+59b613...@syzkaller.appspotmail.com
BUG: KFENCE: use-after-free read in uprobe_cmp kernel/events/uprobes.c:620 [inline]
BUG: KFENCE: use-after-free read in __uprobe_cmp_key kernel/events/uprobes.c:646 [inline]
BUG: KFENCE: use-after-free read in rb_find include/linux/rbtree.h:289 [inline]
BUG: KFENCE: use-after-free read in __find_uprobe kernel/events/uprobes.c:661 [inline]
BUG: KFENCE: use-after-free read in find_uprobe+0x57/0x150 kernel/events/uprobes.c:678
Use-after-free read at 0xffff88823bc8cf68 (in kfence-#69):
uprobe_cmp kernel/events/uprobes.c:620 [inline]
__uprobe_cmp_key kernel/events/uprobes.c:646 [inline]
rb_find include/linux/rbtree.h:289 [inline]
__find_uprobe kernel/events/uprobes.c:661 [inline]
find_uprobe+0x57/0x150 kernel/events/uprobes.c:678
uprobe_apply+0x26/0x130 kernel/events/uprobes.c:1221
uprobe_perf_close+0x41e/0x6f0 kernel/trace/trace_uprobe.c:1
trace_uprobe_register+0x3e7/0x880 kernel/
perf_trace_event_close kernel/
perf_uprobe_
Topics
More
CATEGORIES
General
All categories
TAGS
bugs
gaming
iso
aur
network
All tags
Issues & Assistance
Korozein
hello everyone.
I was checking my error in journalctl and I keep getting this weird one it look like this
Jun 17 08:03:54 cachyos-x8664 kernel:
Jun 17 08:03:54 cachyos-x8664 kernel: BUG: KFENCE: use-after-free read in
_nv000179kms+0x439/0x2a10 [nvidia_modeset]
Jun 17 08:03:54 cachyos-x8664 kernel: Use-after-free read at 0x00000000a818c52f (in
kfence-#243):
Jun 17 08:03:54 cachyos-x8664 kernel: _nv000179kms+0x439/0x2a10 [nvidia_modeset]
Jun 17 08:03:54 cachyos-x8664 kernel: _nv002901kms+0x665/0x9c0 [nvidia_modeset]
Jun 17 08:03:54 cachyos-x8664 kernel: _nv000397kms+0x201/0x430 [nvidia_modeset]
Jun 17 08:03:54 cachyos-x8664 kernel: _nv002900kms+0xeda/0x11b0 [nvidia_modeset]
Jun 17 08:03:54 cachyos-x8664 kernel: _nv003010kms+0x78c/0xd20 [nvidia_modeset]
Jun 17 08:03:54 cachyos-x8664 kernel: nvKmsIoctl+0xf9/0x270 [nvidia_modeset]
Jun 17 08:03:54 cachyos-x8664 kernel: nvkms_ioctl_from_kapi_try_pmlock+0x66/0xc0
Jun 17 08:03:54 cachyos-x8664 kernel: _000023kms+0x566/0xbc0 [nvidia_modeset]
Wiki
Github
Sign Up
Log In
Jun 2025
1 / 6
Jun 2025
Jun 2025

Internals

Internals

> Every **100ms**, KFENCE short-circuits the Slab allocator for a single allocation

Internals

> Every **100ms**, KFENCE short-circuits the Slab allocator for a single allocation

1. slab_alloc_node()

```
1 object = kfence_alloc(s, orig_size, ...);
2 if (unlikely(object))
3     goto out;
4 object = __slab_alloc_node(s, ...);
```

Internals

> Every **100ms**, KFENCE short-circuits the Slab allocator for a single allocation

1. slab_alloc_node()

```
1 object = kfence_alloc(s, orig_size, ...);
2 if (unlikely(object))
3     goto out;
4 object = __slab_alloc_node(s, ...);
```

2. kfence_alloc()

```
1 if (atmic_rd(&kfence_allocation_gate) > 0)
2   return NULL;
3
4 return __kfence_alloc(s, size, flags);
```

Internals

> Every **100ms**, KFENCE short-circuits the Slab allocator for a single allocation

1. slab_alloc_node()

```
1 object = kfence_alloc(s, orig_size, ...);
2 if (unlikely(object))
3     goto out;
4 object = __slab_alloc_node(s, ...);
```

2. kfence_alloc()

```
1 if (atmic_rd(&kfence_allocation_gate) > 0)
2   return NULL;
3
4 return __kfence_alloc(s, size, flags);
```

3. toggle_allocation_gate()

```
1 // INIT_DELAYED_WORK(&kfence_timer, toggle_allocation_gate);
2 atomic_set(&kfence_allocation_gate, -kfence_burst);
3 // ...
4 queue_delayed_work(..., &kfence_timer, msecs_to_jiffies(kfence_sample_interval));
```

Allocation Pool

Allocation Pool

> Served objects are placed in the KFENCE pool: `__kfence_pool`

Allocation Pool

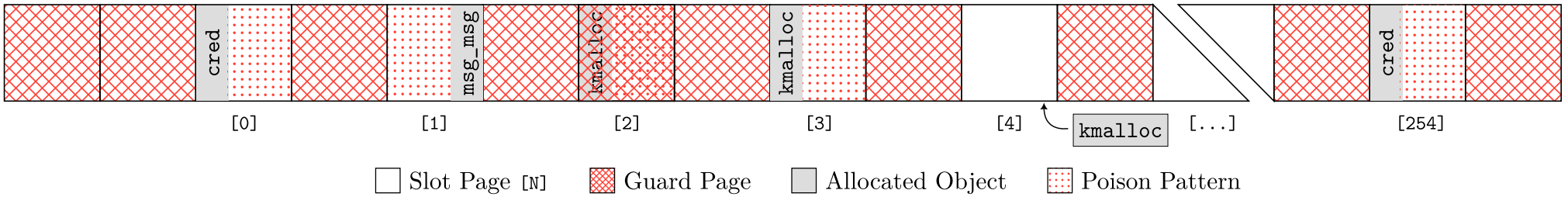
> Served objects are placed in the KFENCE pool: `__kfence_pool`

Objects are placed in **slot pages** surrounded by **guard pages**

Allocation Pool

> Served objects are placed in the KFENCE pool: `__kfence_pool`

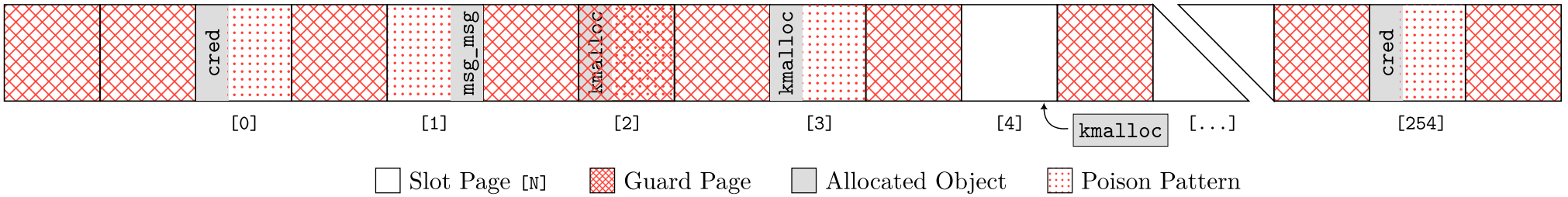
Objects are placed in **slot pages** surrounded by **guard pages**



Allocation Pool

> Served objects are placed in the KFENCE pool: `__kfence_pool`

Objects are placed in **slot pages** surrounded by **guard pages**

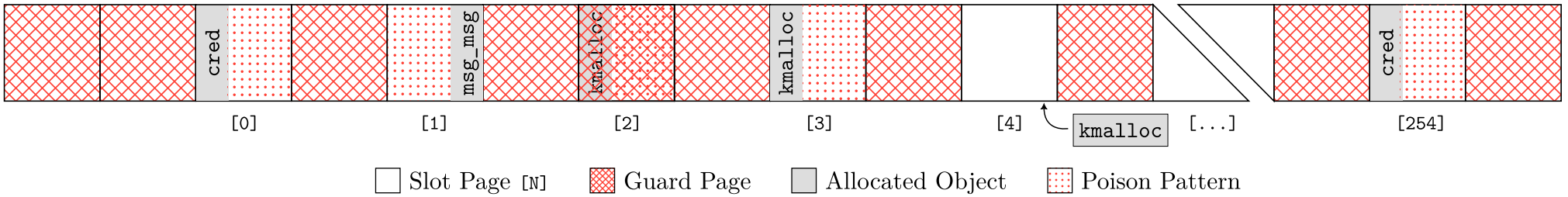


$$(255 + 1) \times 2 \times 4096 = 2\text{MiB}$$

Allocation Pool

> Served objects are placed in the KFENCE pool: `__kfence_pool`

Objects are placed in **slot pages** surrounded by **guard pages**



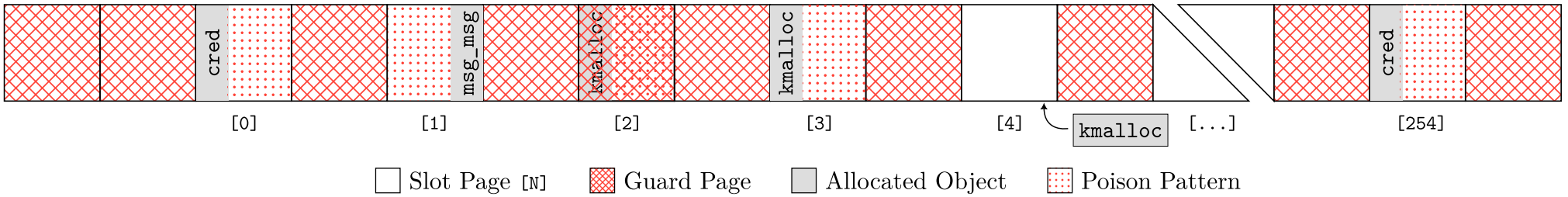
$$(255 + 1) \times 2 \times 4096 = 2\text{MiB}$$

Slot pages hold a **single object** placed at the start or the end of the page

Allocation Pool

> Served objects are placed in the KFENCE pool: `__kfence_pool`

Objects are placed in **slot pages** surrounded by **guard pages**



$$(255 + 1) \times 2 \times 4096 = 2\text{MiB}$$

Slot pages hold a **single object** placed at the start or the end of the page

Guard pages trap on access

```
1 if (protect)
2   set_pte(pte, __pte(pte_val(*pte) & ~_PAGE_PRESENT));
```

Memory Error Detection

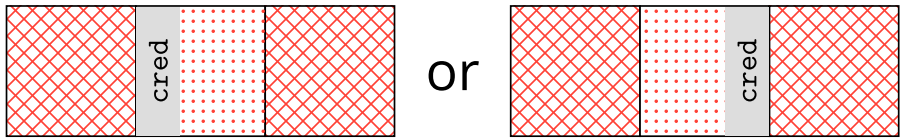
Memory Error Detection

- Spatial and temporal memory satety errors are detected on sampled objects

Memory Error Detection

> Spatial and temporal memory satety errors are detected on sampled objects

Spatial Memory Safety

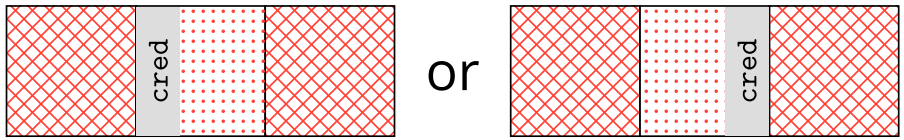


- Randomized start/end placement
- Poison checked on free

Memory Error Detection

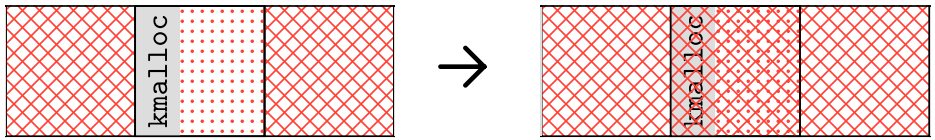
Spatial and temporal memory satety errors are detected on sampled objects

Spatial Memory Safety



- Randomized start/end placement
- Poison checked on free

Temporal Memory Safety

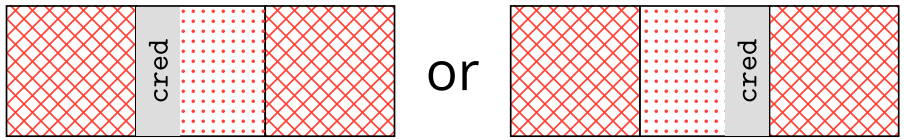


- Freed objects are guarded
- Metadata checks: cache, addr, etc.

Memory Error Detection

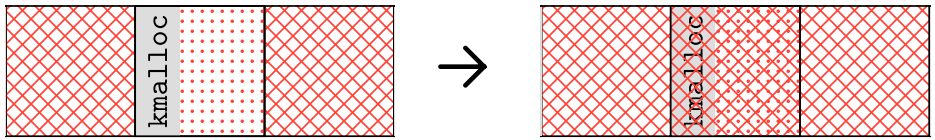
Spatial and temporal memory satety errors are detected on sampled objects

Spatial Memory Safety



- Randomized start/end placement
- Poison checked on free

Temporal Memory Safety



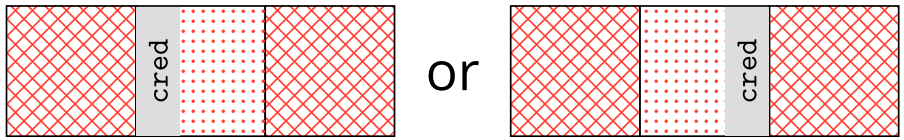
- Freed objects are guarded
- Metadata checks: cache, addr, etc.

Maximizes temporal safety with a **FIFO** freelist:

Memory Error Detection

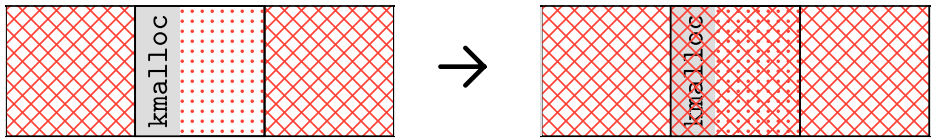
Spatial and temporal memory satety errors are detected on sampled objects

Spatial Memory Safety



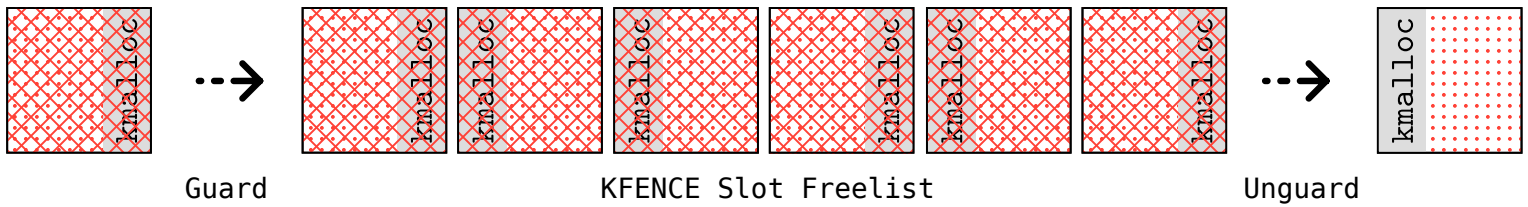
- Randomized start/end placement
- Poison checked on free

Temporal Memory Safety



- Freed objects are guarded
- Metadata checks: cache, addr, etc.

Maximizes temporal safety with a **FIFO** freelist:



Pressure Mode

Pressure Mode

> KFENCE has a special pressure mode, activated at **75% pool occupancy**

Pressure Mode

> KFENCE has a special pressure mode, activated at **75% pool occupancy**

Solves the problem of frequently allocated objects hoarding KFENCE

↳ No coverage on less frequent paths ☹

Pressure Mode

> KFENCE has a special pressure mode, activated at **75% pool occupancy**

Solves the problem of frequently allocated objects hoarding KFENCE

↳ No coverage on less frequent paths ☹

After pressure mode kicks in:

Pressure Mode

> KFENCE has a special pressure mode, activated at **75% pool occupancy**

Solves the problem of frequently allocated objects hoarding KFENCE

↳ No coverage on less frequent paths ☹

After pressure mode kicks in:

```
__alloc_skb+0x72/0x450
iwl_mld_rx_mpdu+0x12a/0x700 [iwlmld]
iwl_pcie_rx_handle_rb.isra.0+0x202/0x430
iwl_pcie_rx_handle+0x1ef/0x650 [iwlwifi]
iwl_pcie_napi_poll_msix+0x3b/0xf0 [iwlwifi]
__napi_poll+0x10f/0x200
net_rx_action+0x34c/0x440
handle_softirqs+0xee/0x330
do_softirq.part.0+0x3b/0x60
__local_bh_enable_ip+0x62/0x70
iwl_pcie_irq_rx_msix_handler+0xdc/0x150 [iwlwifi]
irq_thread_fn+0x25/0x60
irq_thread+0xc7/0x1c0
kthread+0xe4/0x120
ret_from_fork+0x1a1/0x270
ret_from_fork_asm+0x1a/0x30
```

Allocation A

Pressure Mode

> KFENCE has a special pressure mode, activated at **75% pool occupancy**

Solves the problem of frequently allocated objects hoarding KFENCE

↳ No coverage on less frequent paths ☹

After pressure mode kicks in:

```
__alloc_skb+0x72/0x450
iwl_mld_rx_mpdu+0x12a/0x700 [iwlmld]
iwl_pcie_rx_handle_rb.isra.0+0x202/0x430
iwl_pcie_rx_handle+0x1ef/0x650 [iwlwifi]
iwl_pcie_napi_poll_msix+0x3b/0xf0 [iwlwifi]
__napi_poll+0x10f/0x
net_rx_action+0x
handle_soft
do_softi
__local_b
iwl_pcie_irq_rx_msix_handler+0xdc/0x150 [iwlwifi]
irq_thread_fn+0x25/0x60
irq_thread+0xc7/0x1c0
kthread+0xe4/0x120
ret_from_fork+0x1a1/0x270
ret_from_fork_asm+0x1a/0x30
```

fda58d513

Allocation A

Pressure Mode

> KFENCE has a special pressure mode, activated at **75% pool occupancy**

Solves the problem of frequently allocated objects hoarding KFENCE

↳ No coverage on less frequent paths ☹️

After pressure mode kicks in:

```
__alloc_skb+0x72/0x450
iwl_mld_rx_mpdu+0x12a/0x700 [iwlmld]
iwl_pcie_rx_handle_rb.isra.0+0x202/0x430
iwl_pcie_rx_handle+0x1ef/0x650 [iwlwifi]
iwl_pcie_napi_poll_msix+0x3b/0xf0 [iwlwifi]
__napi_poll+0x10f/0x
net_rx_action+0x
handle_softi
do_softi
__local_b
iwl_pcie_irq_rx_msix_handler+0xdc/0x150 [iwlwifi]
irq_thread_fn+0x25/0x60
irq_thread+0xc7/0x1c0
kthread+0xe4/0x120
ret_from_fork+0x1a1/0x270
ret_from_fork_asm+0x1a/0x30
```

fda58d513 ✅

Allocation A

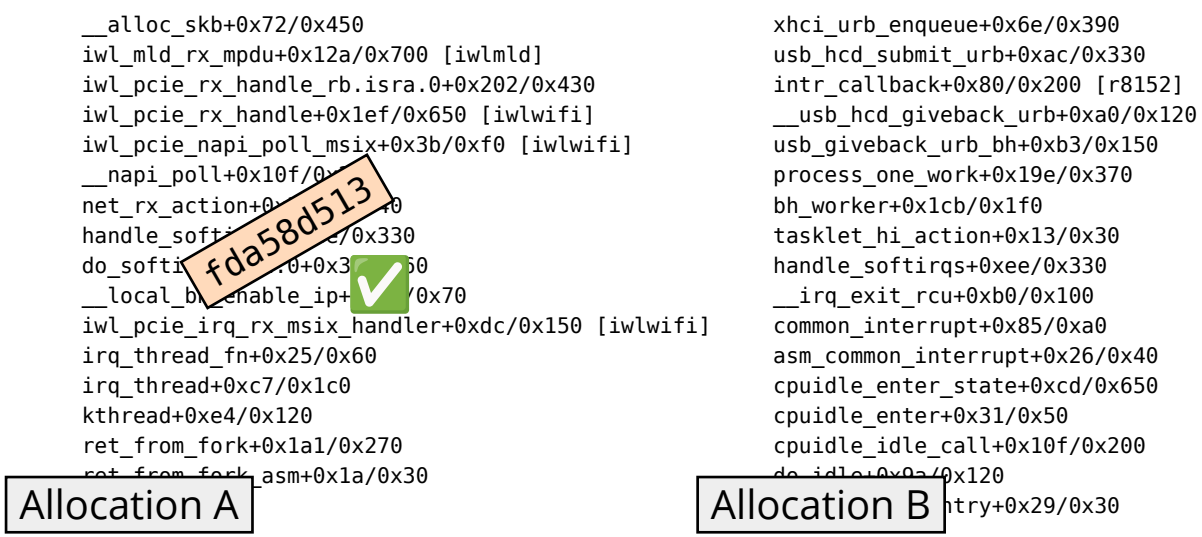
Pressure Mode

> KFENCE has a special pressure mode, activated at **75% pool occupancy**

Solves the problem of frequently allocated objects hoarding KFENCE

↳ No coverage on less frequent paths ☹️

After pressure mode kicks in:



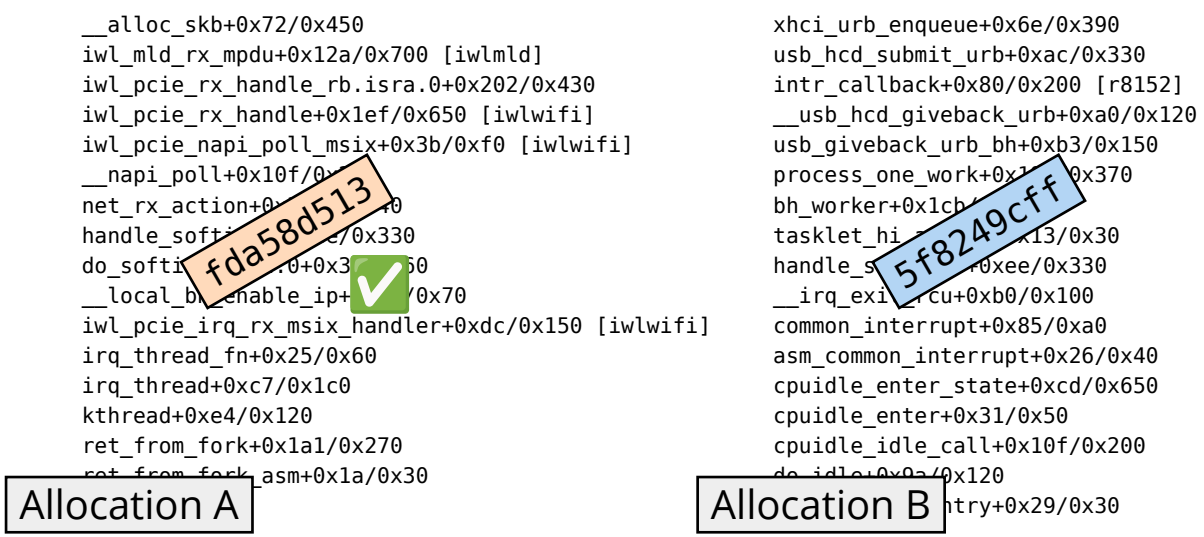
Pressure Mode

> KFENCE has a special pressure mode, activated at **75% pool occupancy**

Solves the problem of frequently allocated objects hoarding KFENCE

↳ No coverage on less frequent paths ☹️

After pressure mode kicks in:



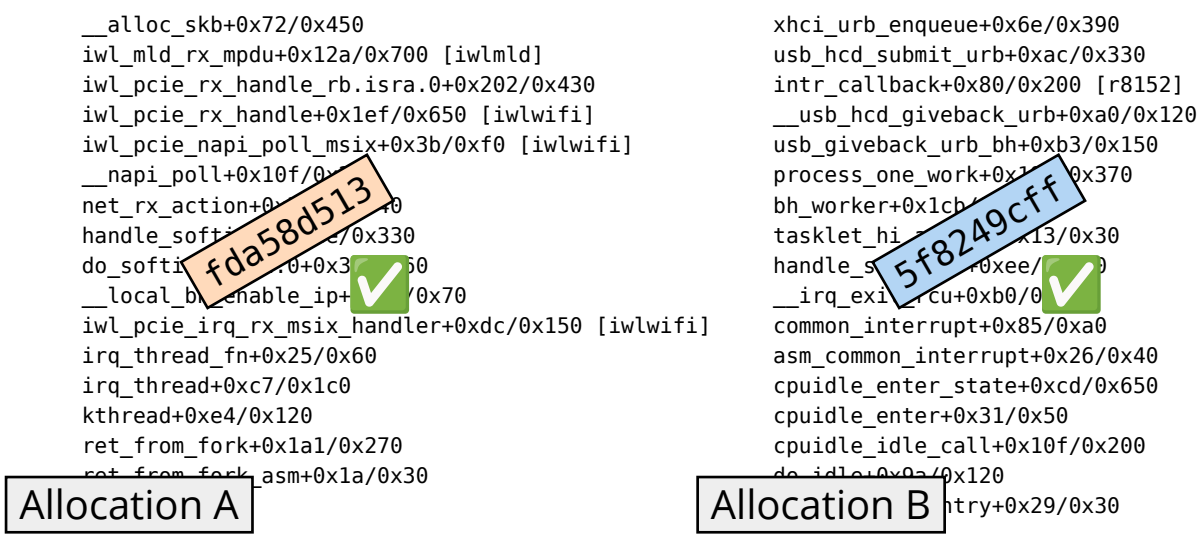
Pressure Mode

> KFENCE has a special pressure mode, activated at **75% pool occupancy**

Solves the problem of frequently allocated objects hoarding KFENCE

↳ No coverage on less frequent paths ☹️

After pressure mode kicks in:



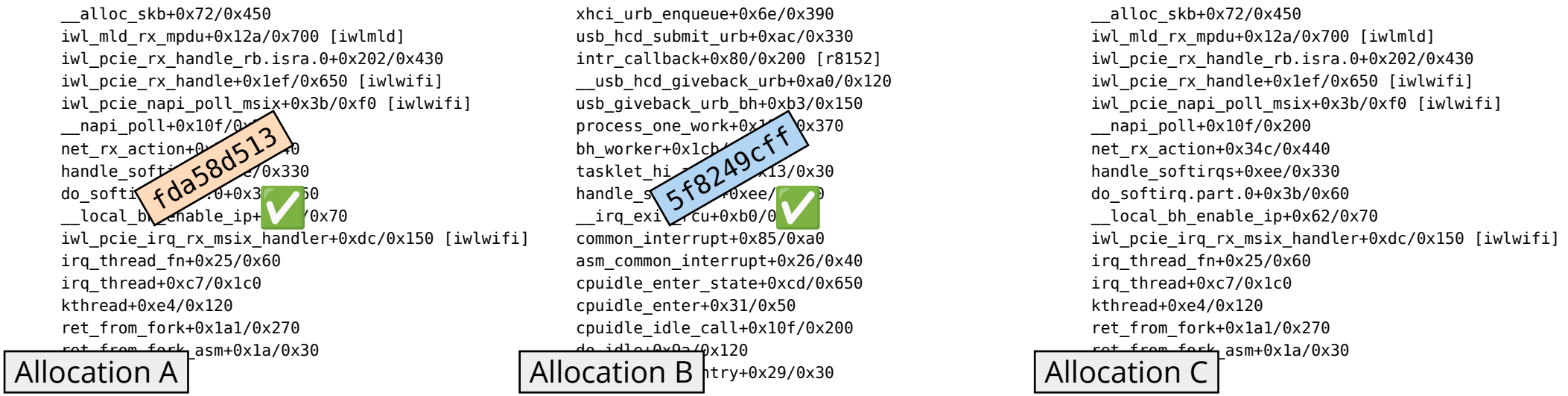
Pressure Mode

> KFENCE has a special pressure mode, activated at **75% pool occupancy**

Solves the problem of frequently allocated objects hoarding KFENCE

↳ No coverage on less frequent paths ☹

After pressure mode kicks in:



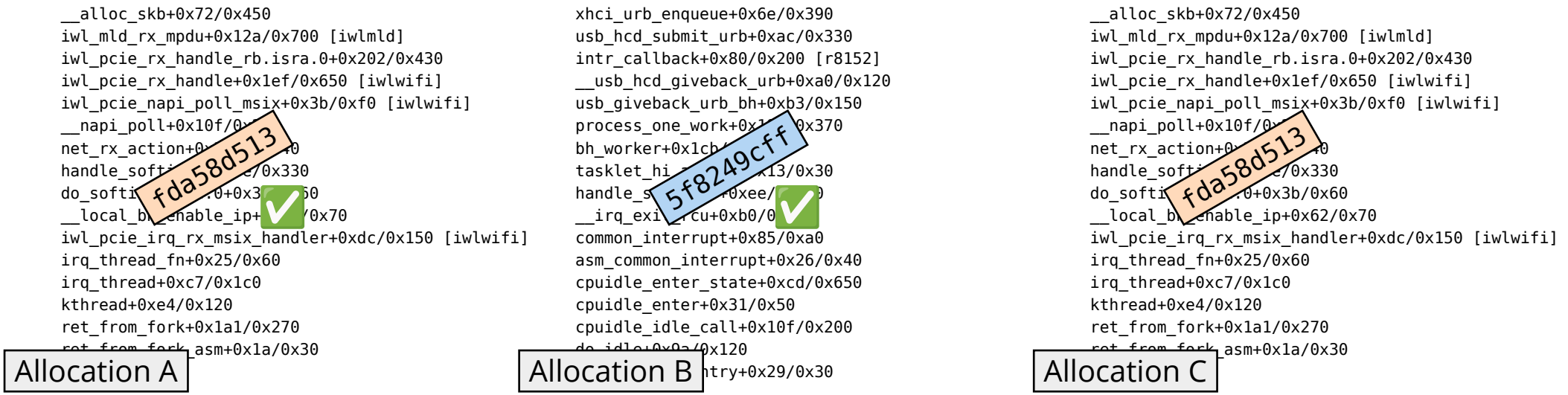
Pressure Mode

> KFENCE has a special pressure mode, activated at **75% pool occupancy**

Solves the problem of frequently allocated objects hoarding KFENCE

↳ No coverage on less frequent paths ☹

After pressure mode kicks in:



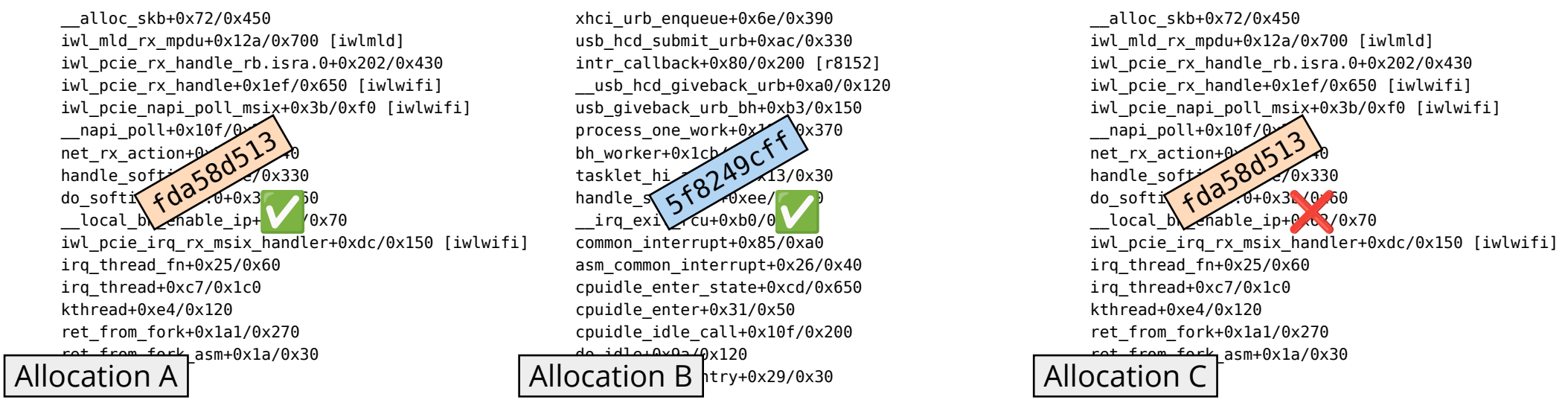
Pressure Mode

> KFENCE has a special pressure mode, activated at **75% pool occupancy**

Solves the problem of frequently allocated objects hoarding KFENCE

↳ No coverage on less frequent paths ☹️

After pressure mode kicks in:



Live Statistics I

Live Statistics I


```
[ecomaikgolf@laptop ~/]$ uptime -p
up 4 days, 13 hours, 41 minutes
```

Live Statistics I


```
[ecomaikgolf@laptop ~/]$ uptime -p
up 4 days, 13 hours, 41 minutes
```

```
[ecomaikgolf@laptop ~/]$ sudo cat /sys/kernel/debug/kfence/stats
enabled: 1
currently allocated: 191
total allocations: 1466277
total frees: 1466086
[...]
skipped allocations (incompatible): 729
skipped allocations (capacity): 0
skipped allocations (covered): 22066
```

Live Statistics I


```
[ecomaikgolf@laptop ~/]$ uptime -p
up 4 days, 13 hours, 41 minutes
```

```
[ecomaikgolf@laptop ~/]$ sudo cat /sys/kernel/debug/kfence/stats
enabled: 1
currently allocated: 191
total allocations: 1466277
total frees: 1466086
[...]
skipped allocations (incompatible): 729
skipped allocations (capacity): 0
skipped allocations (covered): 22066
```

- **1.4M** allocations and frees
- **75%** currently allocated slots
- **1.5%** of total allocations skipped due to pressure mode

Live Statistics II

Live Statistics II

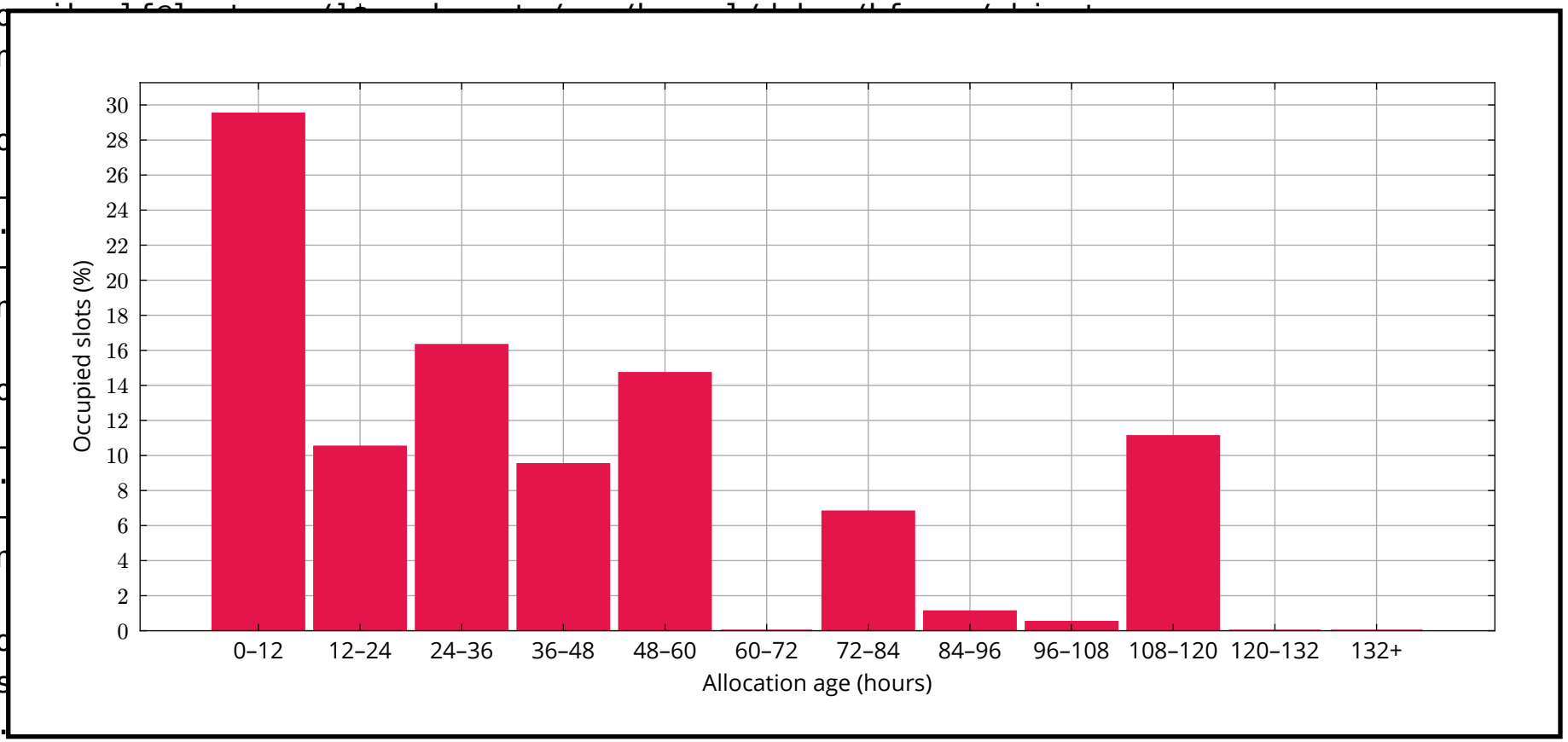

```
[ecomaikgolf@laptop ~/]$ sudo cat /sys/kernel/debug/kfence/objects
kfence-#0: 0x0000000019fd7bdf-0x00000000927e0e5c, size=8, cache=zs_handle

allocated by task 101 on cpu 0 at 224723.303159s (169444.589377s ago):
 zs_malloc+0x5d/0x3d0
 [...]
--------------------------------
kfence-#1: 0x00000000c2abdc15-0x00000000f4996c0d, size=192, cache=vm_area_struct

allocated by task 288050 on cpu 3 at 265381.908716s (128785.983849s ago):
 vm_area_alloc+0x1e/0x80
 [...]
--------------------------------
kfence-#2: 0x00000000ac5cb19a-0x000000002eb918d0, size=576, cache=radix_tree_node

allocated by task 265524 on cpu 7 at 368668.334847s (25499.557727s ago):
 xas_alloc+0xa3/0xd0
 [...]
```

Live Statistics II



Security Analysis



Heap Segregation

Heap Segregation

> KFENCE cannot honor heap segregation with a limited global memory pool

Heap Segregation

> KFENCE cannot honor heap segregation with a limited global memory pool

We employ Slab caches to segregate the heap for security (amongst others):

Heap Segregation

> KFENCE cannot honor heap segregation with a limited global memory pool

We employ Slab caches to segregate the heap for security (amongst others):

`cred`	`kmalloc-*-128`	`msg_msg-96`	`task_struct`

Heap Segregation

> KFENCE cannot honor heap segregation with a limited global memory pool

We employ Slab caches to segregate the heap for security (amongst others):

`cred`	`kmalloc-*-128`	`msg_msg-96`	`task_struct`

Attackers must do in-cache (limited) or cross-cache (probabilistic) attacks

Heap Segregation

> KFENCE cannot honor heap segregation with a limited global memory pool

We employ Slab caches to segregate the heap for security (amongst others):

cred	kmalloc-*-128	msg_msg-96	task_struct

Attackers must do in-cache (limited) or cross-cache (probabilistic) attacks

Contrary, **KFENCE does not perform heap segregation**:

Heap Segregation

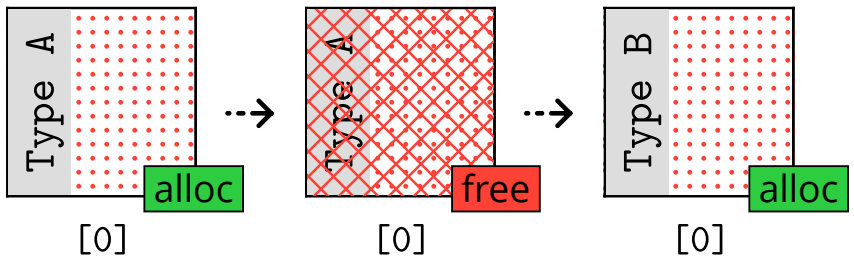
> KFENCE cannot honor heap segregation with a limited global memory pool

We employ Slab caches to segregate the heap for security (amongst others):

cred	kmalloc-*-128	msg_msg-96	task_struct

Attackers must do in-cache (limited) or cross-cache (probabilistic) attacks

Contrary, **KFENCE does not perform heap segregation**:



Heap Segregation

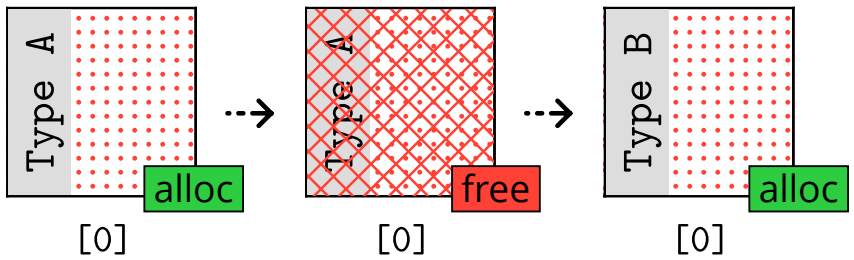
> KFENCE cannot honor heap segregation with a limited global memory pool

We employ Slab caches to segregate the heap for security (amongst others):

cred	kmalloc-*-128	msg_msg-96	task_struct

Attackers must do in-cache (limited) or cross-cache (probabilistic) attacks

Contrary, **KFENCE does not perform heap segregation**:



Furthermore, no size segregation gives attackers have more freedom

Let’s do an experiment

```
# grep kfence /sys/kernel/debug/kfence/objects | head -n 30

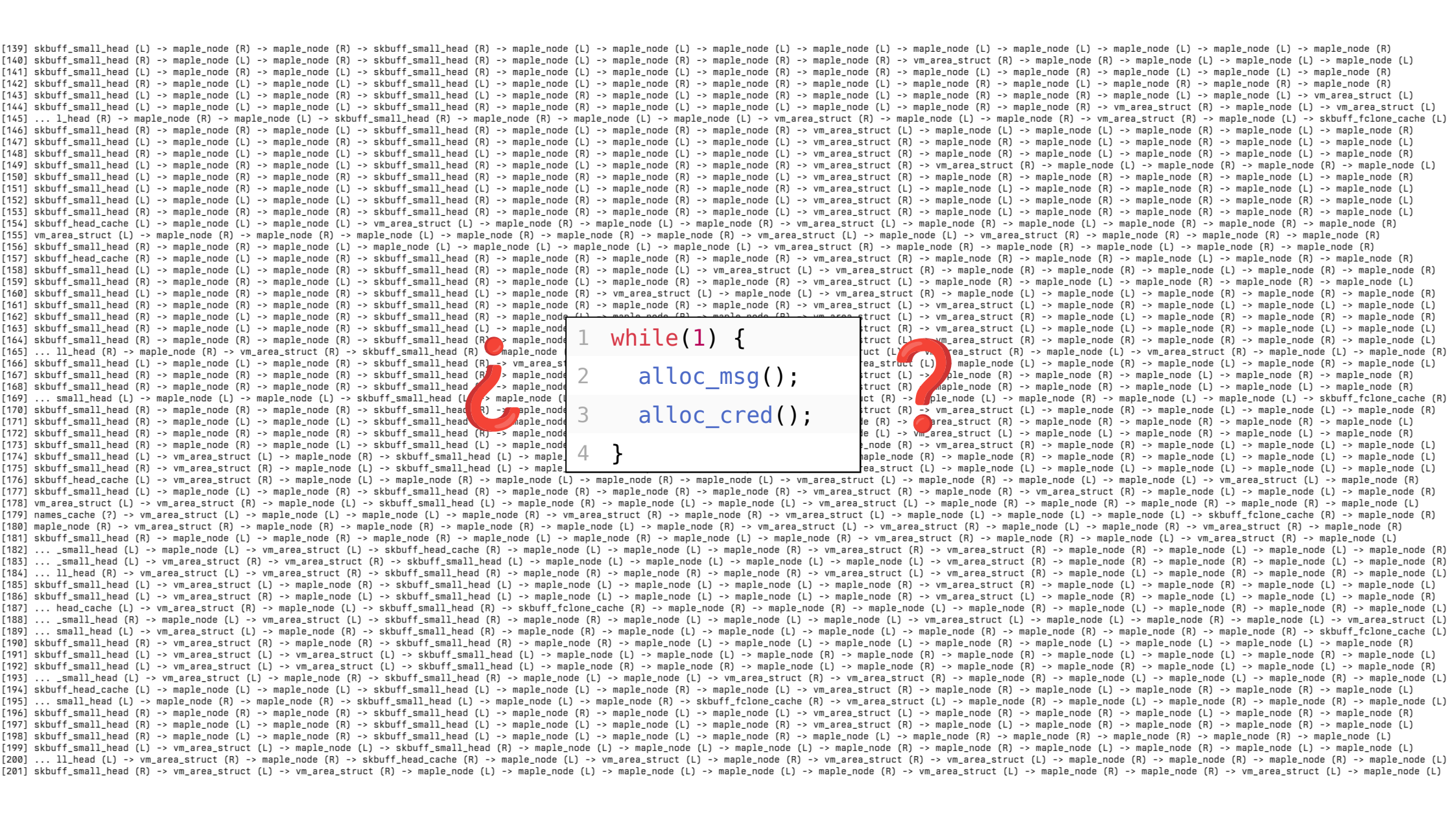
kfence-#0: 0xffffa4203b402000-0xffffa4203b4022bf, size=704, cache=skbuff_small_head
kfence-#1: 0xffffa4203b404d40-0xffffa4203b404fff, size=704, cache=skbuff_small_head
kfence-#2: 0xffffa4203b406000-0xffffa4203b4062bf, size=704, cache=skbuff_small_head
kfence-#3: 0xffffa4203b408d40-0xffffa4203b408fff, size=704, cache=skbuff_small_head
kfence-#4: 0xffffa4203b40a000-0xffffa4203b40afff, size=4096, cache=names_cache
kfence-#5: 0xffffa4203b40c000-0xffffa4203b40c2bf, size=704, cache=skbuff_small_head
kfence-#6: 0xffffa4203b40efc0-0xffffa4203b40efff, size=64, cache=kmalloc-rnd-14-64
kfence-#7: 0xffffa4203b410f00-0xffffa4203b410fdf, size=224, cache=skbuff_head_cache
kfence-#8: 0xffffa4203b412d40-0xffffa4203b412fff, size=704, cache=skbuff_small_head
kfence-#9: 0xffffa4203b414000-0xffffa4203b4142bf, size=704, cache=skbuff_small_head
kfence-#10: 0xffffa4203b416000-0xffffa4203b4162bf, size=704, cache=skbuff_small_head
kfence-#11: 0xffffa4203b418d40-0xffffa4203b418fff, size=704, cache=skbuff_small_head
kfence-#12: 0xffffa4203b41a000-0xffffa4203b41a2bf, size=704, cache=skbuff_small_head
kfence-#13: 0xffffa4203b41cd40-0xffffa4203b41cfff, size=704, cache=skbuff_small_head
kfence-#14: 0xffffa4203b41ed40-0xffffa4203b41efff, size=704, cache=skbuff_small_head
kfence-#15: 0xffffa4203b420d40-0xffffa4203b420fff, size=704, cache=skbuff_small_head
kfence-#16: 0xffffa4203b422d40-0xffffa4203b422fff, size=704, cache=skbuff_small_head
kfence-#17: 0xffffa4203b424d40-0xffffa4203b424fff, size=704, cache=skbuff_small_head
kfence-#18: 0xffffa4203b426d40-0xffffa4203b426fff, size=704, cache=skbuff_small_head
kfence-#19: 0xffffa4203b428d40-0xffffa4203b428fff, size=704, cache=skbuff_small_head
kfence-#20: 0xffffa4203b42a000-0xffffa4203b42a00f, size=16, cache=kmalloc-rnd-14-16
kfence-#21: 0xffffa4203b42cd40-0xffffa4203b42cfff, size=704, cache=skbuff_small_head
kfence-#22: 0xffffa4203b42e000-0xffffa4203b42e2bf, size=704, cache=skbuff_small_head
kfence-#23: 0xffffa4203b430000-0xffffa4203b4302bf, size=704, cache=skbuff_small_head
kfence-#24: 0xffffa4203b432000-0xffffa4203b4322bf, size=704, cache=skbuff_small_head
```

```
# grep kfence /sys/kernel/debug/kfence/objects | head -n 30

kfence-#0: 0xffffa4203b402000-0xffffa4203b4022bf, size=704, cache=skbuff_small_head
kfence-#1: 0xffffa4203b404d40-0xffffa4203b404fff, size=704, cache=skbuff_small_head
kfence-#2: 0xffffa4203b406000-0xffffa4203b4062bf, size=704, cache=skbuff_small_head
kfence-#3: 0xffffa4203b408d40-0xffffa4203b408fff, size=704, cache=skbuff_small_head
kfence-#4: 0xffffa4203b40a000-0xffffa4203b40afff, size=4096, cache=names_cache
kfence-#5: 0xffffa4203b40c000-0xffffa4203b40c2bf, size=704, cache=skbuff_small_head
kfence-#6: 0xffffa4203b40efc0-0xffffa4203b40efff, size=64, cache=kmalloc-rnd-14-64
kfence-#7: 0xffffa4203b410f00-0xffffa4203b410fdf, size=224, cache=skbuff_head_cache
kfence-#8: 0xffffa4203b412d40-0xffffa4203b412fff, size=704, cache=skbuff_small_head
kfence-#9: 0xffffa4203b414000-0xffffa4203b4142bf, size=704, cache=skbuff_small_head
kfence-#10: 0xffffa4203b416000-0xffffa4203b4162bf, size=704, cache=skbuff_small_head
kfence-#11: 0xffffa4203b418d40-0xffffa4203b418fff, size=704, cache=skbuff_small_head
kfence-#12: 0xffffa4203b41a000-0xffffa4203b41a2bf, size=704, cache=skbuff_small_head
kfence-#13: 0xffffa4203b41cd40-0xffffa4203b41cfff, size=704, cache=skbuff_small_head
kfence-#14: 0xffffa4203b41ed40-0xffffa4203b41efff, size=704, cache=skbuff_small_head
kfence-#15: 0xffffa4203b420d40-0xffffa4203b420fff, size=704, cache=skbuff_small_head
kfence-#16: 0xffffa4203b422d40-0xffffa4203b422fff, size=704, cache=skbuff_small_head
kfence-#17: 0xffffa4203b424d40-0xffffa4203b424fff, size=704, cache=skbuff_small_head
kfence-#18: 0xffffa4203b426d40-0xffffa4203b426fff, size=704, cache=skbuff_small_head
kfence-#19: 0xffffa4203b428d40-0xffffa4203b428fff, size=704, cache=skbuff_small_head
kfence-#20: 0xffffa4203b42a000-0xffffa4203b42a00f, size=16, cache=kmalloc-rnd-14-16
kfence-#21: 0xffffa4203b42cd40-0xffffa4203b42cfff, size=704, cache=skbuff_small_head
kfence-#22: 0xffffa4203b42e000-0xffffa4203b42e2bf, size=704, cache=skbuff_small_head
kfence-#23: 0xffffa4203b430000-0xffffa4203b4302bf, size=704, cache=skbuff_small_head
kfence-#24: 0xffffa4203b432000-0xffffa4203b4322bf, size=704, cache=skbuff_small_head
```

[139] vm_area_struct (L)
[140] vm_area_struct (L)
[141] vm_area_struct (L)
[142] vm_area_struct (L)
[143] vm_area_struct (L)
[144] vm_area_struct (R)
[145] vm_area_struct (R)
[146] vm_area_struct (L)
[147] vm_area_struct (L)
[148] vm_area_struct (R)
[149] vm_area_struct (R)
[150] vm_area_struct (R)
[151] vm_area_struct (L)
[152] vm_area_struct (R)
[153] names_cache (?)
[154] vm_area_struct (L)
[155] vm_area_struct (L)
[156] vm_area_struct (R)
[157] vm_area_struct (L)
[158] vm_area_struct (R)
[159] vm_area_struct (R)
[160] vm_area_struct (L)
[161] vm_area_struct (L)
[162] vm_area_struct (L)
[163] vm_area_struct (R)
[164] vm_area_struct (R)
[165] vm_area_struct (L)
[166] vm_area_struct (L)
[167] vm_area_struct (R)
[168] vm_area_struct (R)
[169] vm_area_struct (L)
[170] vm_area_struct (R)
[171] vm_area_struct (R)
[172] vm_area_struct (R)
[173] vm_area_struct (L)
[174] vm_area_struct (L)
[175] vm_area_struct (R)
[176] vm_area_struct (L)
[177] vm_area_struct (L)
[178] vm_area_struct (R)
[179] vm_area_struct (R)
[180] vm_area_struct (L)
[181] vm_area_struct (L)
[182] vm_area_struct (L)
[183] vm_area_struct (R)
[184] vm_area_struct (L)
[185] vm_area_struct (R)
[186] vm_area_struct (L)
[187] vm_area_struct (L)
[188] vm_area_struct (R)
[189] vm_area_struct (R)
[190] vm_area_struct (R)
[191] vm_area_struct (R)
[192] vm_area_struct (R)
[193] vm_area_struct (L)
[194] vm_area_struct (L)
[195] vm_area_struct (L)
[196] vm_area_struct (R)
[197] vm_area_struct (R)
[198] vm_area_struct (R)
[199] vm_area_struct (R)
[200] vm_area_struct (R)
[201] vm_area_struct (R)

[139] skbuff_small_head (L) -> maple_node (R) -> maple_node (R) -> skbuff_small_head (R) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (R)
[140] skbuff_small_head (R) -> maple_node (L) -> maple_node (R) -> skbuff_small_head (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> vm_area_struct (R) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (L)
[141] skbuff_small_head (L) -> maple_node (R) -> maple_node (L) -> skbuff_small_head (R) -> maple_node (L) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (R)
[142] skbuff_small_head (R) -> maple_node (L) -> maple_node (L) -> skbuff_small_head (L) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (R)
[143] skbuff_small_head (L) -> maple_node (L) -> maple_node (R) -> skbuff_small_head (L) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> vm_area_struct (L)
[144] skbuff_small_head (L) -> maple_node (L) -> maple_node (L) -> skbuff_small_head (R) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> vm_area_struct (R) -> maple_node (L) -> vm_area_struct (L)
[145] ... l_head (R) -> maple_node (R) -> maple_node (L) -> skbuff_small_head (R) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> vm_area_struct (R) -> maple_node (L) -> maple_node (R) -> vm_area_struct (R) -> maple_node (L) -> skbuff_fclone_cache (L)
[146] skbuff_small_head (R) -> maple_node (R) -> maple_node (L) -> skbuff_small_head (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> vm_area_struct (L) -> maple_node (L) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> maple_node (R)
[147] skbuff_small_head (L) -> maple_node (L) -> maple_node (R) -> skbuff_small_head (R) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> vm_area_struct (R) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> maple_node (L)
[148] skbuff_small_head (R) -> maple_node (L) -> maple_node (L) -> skbuff_small_head (L) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> vm_area_struct (R) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> maple_node (R)
[149] skbuff_small_head (L) -> maple_node (R) -> maple_node (L) -> skbuff_small_head (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> vm_area_struct (R) -> vm_area_struct (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (L)
[150] skbuff_small_head (L) -> maple_node (R) -> maple_node (R) -> skbuff_small_head (R) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> vm_area_struct (R) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (R)
[151] skbuff_small_head (L) -> maple_node (R) -> maple_node (L) -> skbuff_small_head (L) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> vm_area_struct (L) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (L)
[152] skbuff_small_head (L) -> maple_node (L) -> maple_node (L) -> skbuff_small_head (R) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> vm_area_struct (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (R) -> maple_node (L)
[153] skbuff_small_head (R) -> maple_node (R) -> maple_node (R) -> skbuff_small_head (R) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> vm_area_struct (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (R) -> maple_node (L)
[154] skbuff_head_cache (L) -> maple_node (L) -> maple_node (L) -> vm_area_struct (L) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> vm_area_struct (L) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (R)
[155] vm_area_struct (L) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (R) -> vm_area_struct (L) -> maple_node (L) -> vm_area_struct (R) -> maple_node (R) -> maple_node (R) -> maple_node (R)
[156] skbuff_small_head (R) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> vm_area_struct (R) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (R)
[157] skbuff_head_cache (R) -> maple_node (L) -> maple_node (R) -> skbuff_small_head (R) -> maple_node (R) -> maple_node (R) -> vm_area_struct (R) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (R)
[158] skbuff_small_head (L) -> maple_node (L) -> maple_node (R) -> skbuff_small_head (R) -> maple_node (R) -> maple_node (L) -> vm_area_struct (L) -> vm_area_struct (R) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (R)
[159] skbuff_small_head (R) -> maple_node (R) -> maple_node (L) -> skbuff_small_head (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> vm_area_struct (L) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (L)
[160] skbuff_small_head (L) -> maple_node (R) -> maple_node (R) -> skbuff_small_head (L) -> maple_node (R) -> vm_area_struct (L) -> maple_node (L) -> vm_area_struct (R) -> maple_node (L) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (R)
[161] skbuff_small_head (R) -> maple_node (R) -> maple_node (R) -> skbuff_small_head (R) -> maple_node (R) -> maple_node (R) -> maple_node (R) -> vm_area_struct (L) -> vm_area_struct (L) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (L)
[162] skbuff_small_head (R) -> maple_node (L) -> maple_node (R) -> skbuff_small_head (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> vm_area_struct (L) -> vm_area_struct (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (R)
[163] skbuff_small_head (R) -> maple_node (L) -> maple_node (R) -> skbuff_small_head (L) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> vm_area_struct (R) -> vm_area_struct (L) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> maple_node (L)
[164] skbuff_small_head (R) -> maple_node (R) -> maple_node (R) -> skbuff_small_head (R) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> vm_area_struct (L) -> vm_area_struct (R) -> maple_node (R) -> maple_node (R) -> maple_node (R) -> maple_node (L)
[165] ... ll_head (R) -> maple_node (R) -> vm_area_struct (R) -> skbuff_small_head (R) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> vm_area_struct (L) -> vm_area_struct (R) -> maple_node (L) -> vm_area_struct (R) -> maple_node (L) -> maple_node (R)
[166] skbuff_small_head (L) -> maple_node (L) -> maple_node (R) -> skbuff_small_head (R) -> vm_area_struct (L) -> maple_node (R) -> maple_node (L) -> vm_area_struct (L) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (L)
[167] skbuff_small_head (R) -> maple_node (R) -> maple_node (R) -> skbuff_small_head (R) -> maple_node (R) -> maple_node (R) -> maple_node (R) -> vm_area_struct (L) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (R)
[168] skbuff_small_head (R) -> maple_node (R) -> maple_node (R) -> skbuff_small_head (L) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> vm_area_struct (R) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (R) -> maple_node (R)
[169] ... small_head (L) -> maple_node (L) -> maple_node (L) -> skbuff_small_head (L) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> vm_area_struct (R) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> skbuff_fclone_cache (R)
[170] skbuff_small_head (R) -> maple_node (R) -> maple_node (R) -> skbuff_small_head (R) -> maple_node (R) -> maple_node (R) -> maple_node (R) -> vm_area_struct (R) -> vm_area_struct (L) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (R)
[171] skbuff_small_head (R) -> maple_node (L) -> skbuff_small_head (R) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> vm_area_struct (R) -> maple_node (R) -> maple_node (R) -> maple_node (L)
[172] skbuff_small_head (R) -> maple_node (R) -> maple_node (R) -> skbuff_small_head (R) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> vm_area_struct (L) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> maple_node (R)
[173] skbuff_small_head (R) -> maple_node (R) -> maple_node (L) -> skbuff_small_head (L) -> maple_node (L) -> maple_node (L) -> vm_area_struct (L) -> maple_node (R) -> vm_area_struct (R) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (L)
[174] skbuff_small_head (L) -> vm_area_struct (L) -> maple_node (R) -> skbuff_small_head (L) -> maple_node (R) -> maple_node (R) -> vm_area_struct (L) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (L)
[175] skbuff_small_head (R) -> vm_area_struct (R) -> maple_node (L) -> skbuff_small_head (L) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> vm_area_struct (L) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (L)
[176] skbuff_head_cache (L) -> vm_area_struct (R) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> vm_area_struct (L) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> vm_area_struct (L) -> maple_node (R)
[177] skbuff_small_head (L) -> maple_node (L) -> maple_node (R) -> skbuff_small_head (R) -> maple_node (R) -> maple_node (R) -> maple_node (R) -> vm_area_struct (R) -> maple_node (R) -> vm_area_struct (R) -> maple_node (L) -> maple_node (L) -> maple_node (R)
[178] vm_area_struct (L) -> vm_area_struct (R) -> maple_node (L) -> skbuff_small_head (L) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> vm_area_struct (L) -> maple_node (R) -> maple_node (R) -> maple_node (R) -> maple_node (R) -> maple_node (L)
[179] names_cache (?) -> vm_area_struct (L) -> maple_node (L) -> maple_node (L) -> maple_node (R) -> vm_area_struct (R) -> maple_node (R) -> vm_area_struct (L) -> maple_node (L) -> maple_node (L) -> skbuff_fclone_cache (R) -> maple_node (R)
[180] maple_node (R) -> vm_area_struct (R) -> maple_node (R) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> vm_area_struct (L) -> vm_area_struct (R) -> maple_node (L) -> maple_node (R) -> vm_area_struct (R) -> maple_node (R)
[181] skbuff_small_head (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> vm_area_struct (R) -> maple_node (R) -> maple_node (L) -> vm_area_struct (R) -> maple_node (L)
[182] ... _small_head (L) -> maple_node (L) -> vm_area_struct (L) -> skbuff_head_cache (R) -> maple_node (L) -> maple_node (L) -> maple_node (R) -> vm_area_struct (R) -> vm_area_struct (R) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (R)
[183] ... _small_head (L) -> vm_area_struct (R) -> vm_area_struct (R) -> skbuff_small_head (L) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> vm_area_struct (R) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (R)
[184] ... ll_head (R) -> vm_area_struct (L) -> vm_area_struct (R) -> skbuff_small_head (R) -> maple_node (R) -> maple_node (R) -> maple_node (R) -> vm_area_struct (L) -> vm_area_struct (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (L)
[185] skbuff_small_head (L) -> vm_area_struct (L) -> maple_node (R) -> skbuff_small_head (L) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (R) -> vm_area_struct (R) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> maple_node (L)
[186] skbuff_small_head (L) -> vm_area_struct (R) -> maple_node (L) -> skbuff_small_head (L) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (R) -> vm_area_struct (L) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (R)
[187] ... head_cache (L) -> vm_area_struct (R) -> maple_node (L) -> skbuff_small_head (R) -> skbuff_fclone_cache (R) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (L)
[188] ... _small_head (R) -> maple_node (L) -> vm_area_struct (L) -> skbuff_small_head (R) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> vm_area_struct (L) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> vm_area_struct (L)
[189] ... small_head (L) -> vm_area_struct (L) -> maple_node (R) -> skbuff_small_head (R) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (R) -> maple_node (R) -> skbuff_fclone_cache (L)
[190] skbuff_small_head (R) -> vm_area_struct (R) -> maple_node (R) -> skbuff_small_head (R) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (R)
[191] skbuff_small_head (L) -> vm_area_struct (L) -> vm_area_struct (L) -> skbuff_small_head (L) -> maple_node (L) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (L)
[192] skbuff_small_head (L) -> vm_area_struct (L) -> vm_area_struct (L) -> skbuff_small_head (L) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (R)
[193] ... _small_head (L) -> vm_area_struct (L) -> maple_node (R) -> skbuff_small_head (R) -> maple_node (L) -> maple_node (L) -> vm_area_struct (R) -> vm_area_struct (R) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (R) -> maple_node (L)
[194] skbuff_head_cache (L) -> maple_node (L) -> maple_node (L) -> skbuff_small_head (L) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> vm_area_struct (R) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (L)
[195] ... small_head (L) -> maple_node (R) -> maple_node (R) -> skbuff_small_head (L) -> maple_node (L) -> maple_node (R) -> skbuff_fclone_cache (R) -> vm_area_struct (L) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (L)
[196] skbuff_small_head (R) -> maple_node (R) -> maple_node (R) -> skbuff_small_head (L) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> vm_area_struct (L) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (R)
[197] skbuff_small_head (R) -> maple_node (L) -> maple_node (R) -> skbuff_small_head (L) -> maple_node (L) -> maple_node (L) -> maple_node (R) -> vm_area_struct (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (R) -> maple_node (L)
[198] skbuff_small_head (R) -> maple_node (L) -> maple_node (R) -> skbuff_small_head (L) -> maple_node (L) -> maple_node (L) -> maple_node (R) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (L)
[199] skbuff_small_head (L) -> vm_area_struct (L) -> maple_node (L) -> skbuff_small_head (R) -> maple_node (L) -> maple_node (L) -> maple_node (R) -> maple_node (R) -> maple_node (R) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (L)
[200] ... ll_head (L) -> vm_area_struct (R) -> maple_node (R) -> skbuff_head_cache (R) -> maple_node (L) -> vm_area_struct (L) -> maple_node (R) -> vm_area_struct (R) -> vm_area_struct (L) -> maple_node (R) -> maple_node (R) -> maple_node (R) -> maple_node (L)
[201] skbuff_small_head (R) -> vm_area_struct (L) -> vm_area_struct (R) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (L) -> maple_node (R) -> vm_area_struct (L) -> maple_node (R) -> maple_node (R) -> vm_area_struct (L) -> maple_node (L)


while(1) {
alloc_msg();
alloc_cred();
}

```
[139] msg_msg-64 (R)
[140] pid (R)
[141] msg_msg-64 (L)
[142] msg_msg-64 (L)
[143] cred (R)
[144] msg_msg-64 (R)
[145] msg_msg-64 (L)
[146] msg_msg-64 (R)
[147] msg_msg-64 (R)
[148] sigqueue (R)
[149] msg_msg-64 (R)
[150] msg_msg-64 (R)
[151] msg_msg-64 (L)
[152] msg_msg-64 (R)
[153] msg_msg-64 (R)
[154] msg_msg-64 (R)
[155] msg_msg-64 (R)
[156] msg_msg-64 (L)
[157] msg_msg-64 (L)
[158] msg_msg-64 (R)
[159] msg_msg-64 (R)
[160] msg_msg-64 (L)
[161] msg_msg-64 (R)
[162] msg_msg-64 (R)
[163] msg_msg-64 (R)
[164] msg_msg-64 (R)
[165] msg_msg-64 (L)
[166] msg_msg-64 (L)
[167] kmalloc-rnd-06-32 (R)
[168] msg_msg-64 (R)
[169] msg_msg-64 (R)
[170] msg_msg-64 (R)
[171] msg_msg-64 (L)
[172] msg_msg-64 (L)
[173] msg_msg-64 (R)
[174] msg_msg-64 (L)
[175] msg_msg-64 (R)
[176] msg_msg-64 (R)
[177] msg_msg-64 (L)
[178] msg_msg-64 (R)
[179] msg_msg-64 (R)
[180] msg_msg-64 (L)
[181] msg_msg-64 (L)
[182] msg_msg-64 (L)
[183] msg_msg-64 (R)
[184] sigqueue (L)
[185] msg_msg-64 (R)
[186] msg_msg-64 (L)
[187] msg_msg-64 (R)
[188] msg_msg-64 (R)
[189] vm_area_struct (R)
[190] msg_msg-64 (L)
[191] msg_msg-64 (L)
[192] msg_msg-64 (L)
[193] msg_msg-64 (R)
[194] vm_area_struct (L)
[195] msg_msg-64 (L)
[196] msg_msg-64 (R)
[197] msg_msg-64 (R)
[198] vm_area_struct (L)
[199] msg_msg-64 (L)
[200] sigqueue (R)
[201] sigqueue (L)
```

```c
while(1) {
  alloc_msg();
  alloc_cred();
}
```

[139] maple_node (L) -> pid (L) -> cred (R) -> kmalloc-rnd-06-96 (R) -> cred (L) -> msg_msg-64 (L) -> msg_msg-64 (L) -> cred (R) -> msg_msg-64 (R) -> msg_msg-64 (R) -> maple_node (L) -> msg_msg-64 (L) -> cred (R) -> cred (R)
[140] maple_node (L) -> cred (R) -> cred (L) -> vm_area_struct (L) -> cred (L) -> msg_msg-64 (R) -> msg_msg-64 (R) -> cred (L) -> msg_msg-64 (L) -> msg_msg-64 (R) -> cred (R) -> msg_msg-64 (L) -> cred (L) -> cred (L)
[141] maple_node (R) -> cred (R) -> cred (L) -> cred (R) -> cred (L) -> msg_msg-64 (R) -> msg_msg-64 (R) -> cred (R) -> msg_msg-64 (R) -> msg_msg-64 (L) -> kmalloc-rnd-06-96 (L) -> msg_msg-64 (L) -> cred (L) -> cred (L)
[142] maple_node (L) -> cred (L) -> cred (L) -> kmalloc-rnd-06-96 (R) -> cred (L) -> msg_msg-64 (R) -> msg_msg-64 (R) -> vm_area_struct (L) -> msg_msg-64 (L) -> msg_msg-64 (L) -> vm_area_struct (R) -> msg_msg-64 (R) -> cred (R) -> cred (R)
[143] maple_node (R) -> maple_node (L) -> cred (R) -> sigqueue (L) -> cred (R) -> msg_msg-64 (L) -> msg_msg-64 (R) -> cred (R) -> msg_msg-64 (R) -> msg_msg-64 (L) -> vm_area_struct (L) -> kmalloc-rnd-10-8 (R) -> cred (R) -> cred (L)
[144] maple_node (L) -> cred (L) -> cred (L) -> cred (L) -> cred (R) -> msg_msg-64 (L) -> msg_msg-64 (R) -> cred (R) -> msg_msg-64 (L) -> msg_msg-64 (L) -> cred (L) -> msg_msg-64 (R) -> cred (L) -> cred (R)
[145] maple_node (R) -> cred (R) -> cred (L) -> kmalloc-rnd-06-128 (L) -> cred (L) -> msg_msg-64 (L) -> msg_msg-64 (L) -> cred (L) -> msg_msg-64 (R) -> msg_msg-64 (R) -> cred (L) -> kmalloc-rnd-10-8 (R) -> cred (R) -> maple_node (L)
[146] maple_node (R) -> cred (R) -> cred (R) -> cred (R) -> msg_msg-64 (R) -> msg_msg-64 (L) -> cred (L) -> msg_msg-64 (L) -> msg_msg-64 (L) -> cred (R) -> msg_msg-64 (L) -> vm_area_struct (R) -> vm_area_struct (R)
[147] maple_node (R) -> vm_area_struct (R) -> vm_area_struct (L) -> cred (L) -> cred (L) -> msg_msg-64 (R) -> msg_msg-64 (L) -> cred (R) -> msg_msg-64 (R) -> maple_node (R) -> cred (L) -> msg_msg-64 (R) -> cred (L) -> cred (L)
[148] vm_area_struct (R) -> cred (L) -> cred (L) -> cred (R) -> cred (L) -> msg_msg-64 (R) -> msg_msg-64 (L) -> cred (L) -> msg_msg-64 (L) -> vm_area_struct (R) -> vm_area_struct (R) -> msg_msg-64 (L) -> cred (R) -> cred (L)
[149] maple_node (R) -> fs_cache (L) -> cred (L) -> pid (R) -> cred (R) -> msg_msg-64 (L) -> msg_msg-64 (L) -> cred (R) -> msg_msg-64 (L) -> vm_area_struct (R) -> pid (R) -> msg_msg-64 (L) -> cred (R) -> cred (L)
[150] maple_node (L) -> cred (L) -> cred (R) -> cred (R) -> mm_struct (R) -> msg_msg-64 (L) -> msg_msg-64 (L) -> sigqueue (L) -> vm_area_struct (L) -> skbuff_fclone_cache (R) -> cred (L) -> msg_msg-64 (R) -> cred (L) -> cred (R)
[151] maple_node (L) -> skbuff_small_head (R) -> anon_vma (L) -> cred (L) -> vm_area_struct (R) -> cred (R) -> msg_msg-64 (R) -> maple_node (R) -> msg_msg-64 (R) -> cred (L) -> msg_msg-64 (R) -> cred (R) -> cred (R)
[152] maple_node (R) -> cred (R) -> cred (L) -> cred (R) -> cred (L) -> msg_msg-64 (L) -> sigqueue (R) -> skbuff_fclone_cache (L) -> msg_msg-64 (R) -> cred (R) -> msg_msg-64 (R) -> cred (L) -> cred (L)
[153] maple_node (R) -> vm_area_struct (L) -> cred (L) -> fs_cache (R) -> cred (L) -> cred (L) -> msg_msg-64 (R) -> cred (L) -> msg_msg-64 (R) -> msg_msg-64 (R) -> cred (L) -> msg_msg-64 (L) -> cred (L) -> cred (L)
[154] maple_node (L) -> cred (R) -> cred (L) -> cred (R) -> cred (L) -> cred (L) -> msg_msg-64 (R) -> cred (R) -> msg_msg-64 (L) -> kmalloc-rnd-10-8 (R) -> cred (L) -> msg_msg-64 (R) -> files_cache (L) -> sigqueue (R)
[155] maple_node (R) -> files_cache (R) -> mm_struct (R) -> cred (R) -> cred (L) -> cred (R) -> msg_msg-64 (L) -> vm_area_struct (R) -> msg_msg-64 (L) -> msg_msg-64 (R) -> maple_node (L) -> msg_msg-64 (R) -> cred (L) -> cred (L)
[156] maple_node (L) -> cred (L) -> cred (L) -> cred (L) -> cred (R) -> cred (R) -> maple_node (L) -> cred (R) -> msg_msg-64 (R) -> msg_msg-64 (R) -> vm_area_struct (L) -> msg_msg-64 (L) -> msg_msg-64 (R) -> cred (L)
[157] maple_node (L) -> sigqueue (L) -> cred (L) -> cred (L) -> cred (R) -> cred (R) -> msg_msg-64 (L) -> vm_area_struct (L) -> msg_msg-64 (L) -> msg_msg-64 (L) -> cred (R) -> msg_msg-64 (R) -> msg_msg-64 (L) -> cred (L)
[158] maple_node (R) -> sigqueue (L) -> cred (R) -> cred (L) -> vm_area_struct (R) -> pid (R) -> msg_msg-64 (R) -> sigqueue (R) -> msg_msg-64 (R) -> msg_msg-64 (L) -> cred (L) -> vm_area_struct (R) -> msg_msg-64 (L) -> cred (R)
[159] ... -> skbuff_small_head (R) -> msg_msg-64 (R) -> mm_struct (L) -> vm_area_struct (L) -> cred (L) -> kmalloc-rnd-10-8 (R) -> maple_node (R) -> kmalloc-rnd-10-8 (L) -> msg_msg-64 (R) -> sigqueue (R) -> vm_area_struct (L) -> msg_msg-64 (R) -> fs_cache (R)
[160] maple_node (L) -> sigqueue (L) -> msg_msg-64 (R) -> cred (L) -> cred (L) -> vm_area_struct (L) -> msg_msg-64 (R) -> cred (R) -> msg_msg-64 (L) -> msg_msg-64 (L) -> cred (L) -> msg_msg-64 (L) -> msg_msg-64 (L) -> cred (R)
[161] maple_node (R) -> cred (L) -> msg_msg-64 (L) -> cred (L) -> maple_node (L) -> cred (L) -> msg_msg-64 (R) -> cred (L) -> sigqueue (L) -> msg_msg-64 (R) -> cred (R) -> msg_msg-64 (R) -> kmalloc-rnd-10-8 (L) -> cred (L)
[162] maple_node (L) -> cred (R) -> msg_msg-64 (R) -> cred (R) -> kmalloc-rnd-06-96 (R) -> cred (L) -> kmalloc-rnd-10-8 (R) -> cred (L) -> msg_msg-64 (L) -> msg_msg-64 (L) -> cred (R) -> msg_msg-64 (R) -> kmalloc-rnd-10-8 (R) -> cred (L)
[163] vm_area_struct (R) -> pid (R) -> msg_msg-64 (R) -> kmalloc-rnd-06-96 (R) -> cred (L) -> kmalloc-rnd-06-96 (L) -> msg_msg-64 (L) -> cred (R) -> msg_msg-64 (L) -> msg_msg-64 (L) -> vm_area_struct (R) -> msg_msg-64 (R) -> kmalloc-rnd-10-8 (R) -> cred (L)
[164] vm_area_struct (L) -> cred (L) -> msg_msg-64 (R) -> maple_node (R) -> cred (L) -> cred (L) -> msg_msg-64 (L) -> cred (L) -> msg_msg-64 (R) -> msg_msg-64 (L) -> maple_node (R) -> msg_msg-64 (L) -> msg_msg-64 (L) -> pid (L)
[165] vm_area_struct (L) -> sigqueue (R) -> maple_node (L) -> kmalloc-rnd-06-96 (R) -> cred (R) -> cred (L) -> msg_msg-64 (L) -> cred (R) -> kmalloc-rnd-10-8 (R) -> kmalloc-rnd-10-8 (L) -> cred (L) -> msg_msg-64 (L) -> msg_msg-64 (R) -> cred (R)
[166] vm_area_struct (L) -> cred (L) -> vm_area_struct (R) -> cred (R) -> cred (R) -> cred (L) -> msg_msg-64 (R) -> cred (L) -> msg_msg-64 (L) -> msg_msg-64 (R) -> sigqueue (L) -> msg_msg-64 (R) -> msg_msg-64 (L) -> signal_cache (L)
[167] vm_area_struct (L) -> maple_node (R) -> msg_msg-64 (L) -> cred (R) -> cred (L) -> sigqueue (L) -> msg_msg-64 (R) -> sigqueue (L) -> msg_msg-64 (R) -> msg_msg-64 (L) -> cred (R) -> msg_msg-64 (R) -> msg_msg-64 (R) -> cred (L)
[168] vm_area_struct (R) -> vm_area_struct (L) -> vm_area_struct (R) -> cred (R) -> cred (L) -> cred (R) -> kmalloc-rnd-10-8 (R) -> cred (L) -> msg_msg-64 (R) -> msg_msg-64 (R) -> cred (L) -> msg_msg-64 (L) -> kmalloc-rnd-10-8 (R) -> vm_area_struct (R)
[169] vm_area_struct (R) -> cred (L) -> skbuff_fclone_cache (L) -> cred (R) -> mm_struct (L) -> cred (L) -> msg_msg-64 (R) -> cred (R) -> kmalloc-rnd-10-8 (R) -> msg_msg-64 (R) -> vm_area_struct (L) -> msg_msg-64 (L) -> msg_msg-64 (L) -> maple_node (R)
[170] vm_area_struct (L) -> cred (L) -> msg_msg-64 (R) -> cred (L) -> sigqueue (R) -> cred (L) -> msg_msg-64 (R) -> cred (L) -> msg_msg-64 (L) -> msg_msg-64 (R) -> cred (R) -> msg_msg-64 (L) -> msg_msg-64 (R) -> kmalloc-rnd-06-96 (R)
[171] vm_area_struct (L) -> skbuff_fclone_cache (L) -> msg_msg-64 (R) -> cred (R) -> cred (R) -> cred (R) -> msg_msg-64 (R) -> msg_msg-64 (L) -> msg_msg-64 (R) -> msg_msg-64 (R) -> pid (L) -> msg_msg-64 (L) -> msg_msg-64 (R) -> cred (L)
[172] vm_area_struct (L) -> cred (R) -> msg_msg-64 (R) -> cred (R) -> cred (L) -> cred (R) -> msg_msg-64 (L) -> kmalloc-rnd-06-96 (R) -> msg_msg-64 (L) -> msg_msg-64 (L) -> cred (R) -> msg_msg-64 (R) -> msg_msg-64 (L) -> cred (L)
[173] vm_area_struct (R) -> cred (R) -> msg_msg-64 (R) -> cred (R) -> vm_area_struct (L) -> cred (R) -> msg_msg-64 (R) -> pid (L) -> msg_msg-64 (R) -> msg_msg-64 (L) -> cred (L) -> msg_msg-64 (L) -> msg_msg-64 (R) -> sigqueue (R)
[174] skbuff_small_head (L) -> cred (L) -> msg_msg-64 (R) -> cred (L) -> cred (R) -> sigqueue (L) -> msg_msg-64 (R) -> cred (R) -> vm_area_struct (R) -> msg_msg-64 (L) -> cred (L) -> msg_msg-64 (R) -> msg_msg-64 (L) -> cred (R)
[175] skbuff_head_cache (L) -> cred (L) -> msg_msg-64 (L) -> vm_area_struct (L) -> cred (L) -> cred (L) -> msg_msg-64 (R) -> sigqueue (L) -> msg_msg-64 (R) -> msg_msg-64 (R) -> cred (R) -> msg_msg-64 (R) -> cred (R) -> cred (R)
[176] vm_area_struct (L) -> cred (R) -> msg_msg-64 (L) -> cred (R) -> cred (L) -> cred (R) -> msg_msg-64 (L) -> sigqueue (R) -> msg_msg-64 (L) -> msg_msg-64 (L) -> cred (R) -> msg_msg-64 (L) -> msg_msg-64 (L) -> cred (L)
[177] kmalloc-rnd-06-96 (L) -> cred (L) -> msg_msg-64 (R) -> cred (R) -> cred (R) -> cred (R) -> msg_msg-64 (R) -> vm_area_struct (R) -> skbuff_fclone_cache (R) -> msg_msg-64 (L) -> vm_area_struct (R) -> msg_msg-64 (R) -> msg_msg-64 (L) -> sigqueue (L)
[178] kmalloc-rnd-06-96 (R) -> vm_area_struct (L) -> msg_msg-64 (L) -> cred (R) -> sigqueue (R) -> cred (L) -> msg_msg-64 (L) -> maple_node (R) -> msg_msg-64 (R) -> msg_msg-64 (L) -> cred (R) -> msg_msg-64 (R) -> msg_msg-64 (L) -> cred (L)
[179] sigqueue (L) -> cred (R) -> kmalloc-rnd-10-8 (R) -> cred (R) -> cred (R) -> cred (L) -> msg_msg-64 (L) -> cred (L) -> msg_msg-64 (R) -> msg_msg-64 (R) -> cred (L) -> msg_msg-64 (R) -> msg_msg-64 (R) -> cred (L)
[180] kmalloc-rnd-06-96 (L) -> maple_node (R) -> msg_msg-64 (L) -> cred (L) -> cred (L) -> cred (R) -> vm_area_struct (R) -> cred (L) -> msg_msg-64 (L) -> sigqueue (R) -> cred (R) -> msg_msg-64 (L) -> kmalloc-rnd-10-8 (L) -> cred (R)
[181] sigqueue (L) -> sigqueue (L) -> msg_msg-64 (L) -> cred (L) -> cred (R) -> cred (L) -> msg_msg-64 (R) -> cred (R) -> msg_msg-64 (L) -> msg_msg-64 (R) -> cred (L) -> msg_msg-64 (L) -> maple_node (R) -> sigqueue (R)
[182] cred (R) -> cred (L) -> msg_msg-64 (R) -> cred (L) -> cred (L) -> vm_area_struct (L) -> msg_msg-64 (R) -> cred (L) -> msg_msg-64 (R) -> msg_msg-64 (L) -> fs_cache (L) -> msg_msg-64 (R) -> maple_node (L) -> cred (R)
[183] kmalloc-rnd-06-96 (L) -> cred (L) -> msg_msg-64 (L) -> cred (L) -> cred (L) -> kmalloc-rnd-06-96 (L) -> msg_msg-64 (R) -> cred (R) -> msg_msg-64 (R) -> msg_msg-64 (L) -> cred (L) -> msg_msg-64 (R) -> msg_msg-64 (R) -> cred (R)
[184] sigqueue (L) -> cred (R) -> msg_msg-64 (R) -> cred (R) -> cred (L) -> cred (L) -> msg_msg-64 (R) -> cred (R) -> msg_msg-64 (L) -> msg_msg-64 (L) -> cred (R) -> vm_area_struct (R) -> msg_msg-64 (L) -> pid (L)
[185] kmalloc-rnd-06-96 (L) -> cred (R) -> msg_msg-64 (R) -> cred (L) -> maple_node (L) -> sigqueue (L) -> msg_msg-64 (R) -> cred (R) -> msg_msg-64 (L) -> msg_msg-64 (R) -> vm_area_struct (L) -> msg_msg-64 (R) -> msg_msg-64 (R) -> cred (L)
[186] kmalloc-rnd-06-96 (R) -> cred (L) -> msg_msg-64 (R) -> cred (R) -> vm_area_struct (L) -> msg_msg-64 (R) -> cred (L) -> msg_msg-64 (R) -> msg_msg-64 (R) -> kmalloc-rnd-06-96 (L) -> msg_msg-64 (R) -> msg_msg-64 (R) -> cred (L)
[187] msg_msg-64 (R) -> cred (R) -> msg_msg-64 (R) -> maple_node (L) -> skbuff_fclone_cache (R) -> cred (L) -> msg_msg-64 (R) -> cred (R) -> msg_msg-64 (R) -> kmalloc-rnd-10-8 (R) -> pid (L) -> msg_msg-64 (L) -> msg_msg-64 (R) -> cred (L)
[188] msg_msg-64 (L) -> cred (R) -> msg_msg-64 (L) -> vm_area_struct (R) -> cred (L) -> cred (L) -> msg_msg-64 (L) -> cred (R) -> msg_msg-64 (R) -> msg_msg-64 (R) -> cred (L) -> msg_msg-64 (L) -> msg_msg-64 (L) -> cred (L)
[189] kmalloc-rnd-10-8 (L) -> cred (R) -> msg_msg-64 (R) -> cred (R) -> pid (L) -> cred (R) -> kmalloc-rnd-10-8 (L) -> msg_msg-64 (L) -> msg_msg-64 (R) -> msg_msg-64 (L) -> cred (R) -> msg_msg-64 (R) -> msg_msg-64 (L) -> cred (R)
[190] msg_msg-64 (R) -> cred (R) -> cred (R) -> kmalloc-rnd-06-96 (L) -> cred (L) -> vm_area_struct (R) -> msg_msg-64 (L) -> cred (L) -> msg_msg-64 (L) -> msg_msg-64 (L) -> cred (L) -> msg_msg-64 (R) -> msg_msg-64 (R) -> cred (L)
[191] msg_msg-64 (L) -> cred (R) -> vm_area_struct (L) -> cred (L) -> cred (R) -> cred (L) -> msg_msg-64 (L) -> cred (R) -> msg_msg-64 (L) -> msg_msg-64 (L) -> cred (L) -> sigqueue (R) -> msg_msg-64 (L) ->
[192] msg_msg-64 (R) -> kmalloc-rnd-06-96 (R) -> skbuff_fclone_cache (R) -> cred (R) -> vm_area_struct (L) -> cred (R) -> kmalloc-rnd-10-8 (L) -> sigqueue (L) -> msg_msg-64 (R) -> msg_msg-64 (R) -> cred (R
[193] msg_msg-64 (R) -> cred (L) -> cred (L) -> cred (R) -> cred (L) -> msg_msg-64 (R) -> maple_node (L) -> msg_msg-64 (L) -> msg_msg-64 (L) -> cred (R) -> cred (L) -> msg_msg-64 (L) -> maple_n
[194] msg_msg-64 (R) -> sigqueue (L) -> cred (R) -> vm_area_struct (L) -> sigqueue (R) -> cred (R) -> kmalloc-rnd-10-8 (R) -> cred (R) -> msg_msg-64 (L) -> msg_msg-64 (L) -> maple_node (L) -> cred (L) -> k
[195] msg_msg-64 (L) -> cred (L) -> cred (L) -> cred (R) -> cred (L) -> cred (L) -> msg_msg-64 (R) -> cred (L) -> msg_msg-64 (R) -> msg_msg-64 (L) -> cred (L) -> vm_area_struct (R) -> msg_msg-64 (L) -> kma
[196] msg_msg-64 (R) -> cred (L) -> cred (R) -> cred (R) -> sigqueue (R) -> cred (R) -> msg_msg-64 (R) -> msg_msg-64 (L) -> msg_msg-64 (R) -> msg_msg-64 (L) -> vm_area_struct (R) -> cred (L) -> msg_msg-64
[197] msg_msg-64 (R) -> sigqueue (L) -> cred (L) -> cred (R) -> sigqueue (R) -> cred (L) -> msg_msg-64 (R) -> msg_msg-64 (L) -> msg_msg-64 (R) -> msg_msg-64 (R) -> cred (R) -> cred (L) -> kmalloc-rnd-10-8
[198] maple_node (R) -> cred (R) -> cred (R) -> cred (L) -> cred (L) -> cred (L) -> msg_msg-64 (R) -> msg_msg-64 (L) -> maple_node (R) -> msg_msg-64 (R) -> cred (R) -> pid (R) -> msg_msg-64 (L) -> cred (L)
[199] vm_area_struct (L) -> cred (L) -> cred (L) -> vm_area_struct (L) -> cred (R) -> cred (R) -> msg_msg-64 (R) -> msg_msg-64 (L) -> vm_area_struct (R) -> vm_area_struct (L) -> sigqueue (R) -> cred (R) ->
[200] kmalloc-rnd-10-8 (R) -> cred (L) -> mm_struct (R) -> cred (R) -> cred (R) -> cred (R) -> msg_msg-64 (L) -> kmalloc-rnd-10-8 (R) -> vm_area_struct (L) -> msg_msg-64 (R) -> cred (R) -> cred (L) -> msg_
[201] msg_msg-64 (L) -> cred (L) -> cred (R) -> cred (R) -> maple_node (L) -> vm_area_struct (L) -> msg_msg-64 (R) -> msg_msg-64 (L) -> msg_msg-64 (L) -> msg_msg-64 (L) -> cred (R) -> msg_msg-64 (L) -> msg_

```c
while(1) {
    alloc_msg();
    alloc_cred();
}
```

SLAB_VIRTUAL

SLAB_VIRTUAL

> KFENCE does not honor `SLAB_VIRTUAL` hardened segregation guarantees

SLAB_VIRTUAL

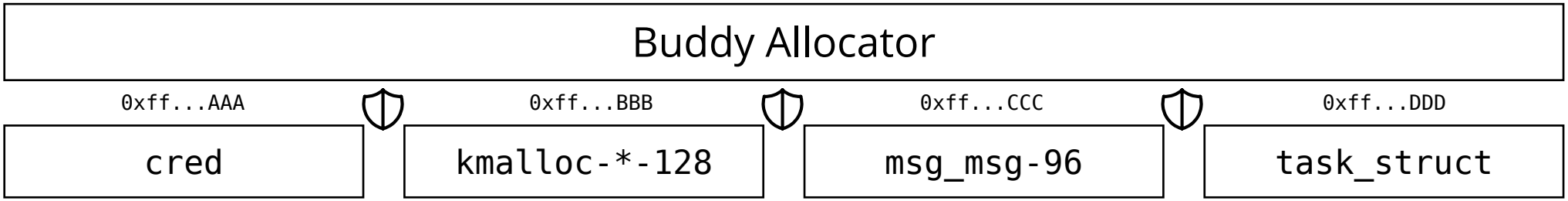
> KFENCE does not honor SLAB_VIRTUAL hardened segregation guarantees

SLAB_VIRTUAL defends against cross-cache attacks by partitioning the VA space:

SLAB_VIRTUAL

> KFENCE does not honor SLAB_VIRTUAL hardened segregation guarantees

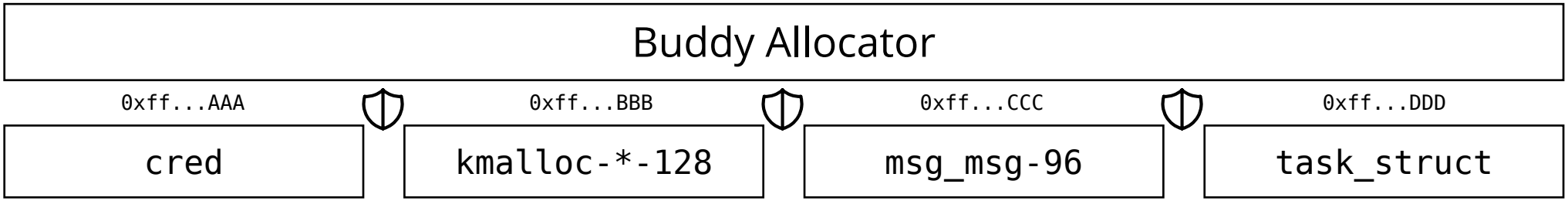
SLAB_VIRTUAL defends against cross-cache attacks by partitioning the VA space:



SLAB_VIRTUAL

> KFENCE does not honor SLAB_VIRTUAL hardened segregation guarantees

SLAB_VIRTUAL defends against cross-cache attacks by partitioning the VA space:

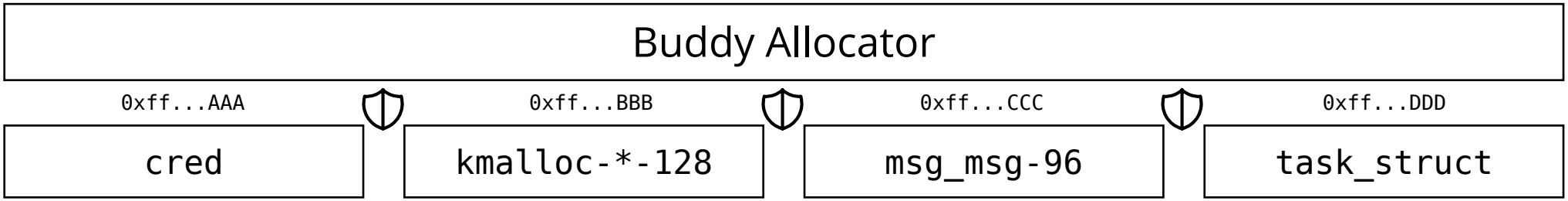


Cross-cache **use-after-free** exploitation can be **revived** under KFENCE

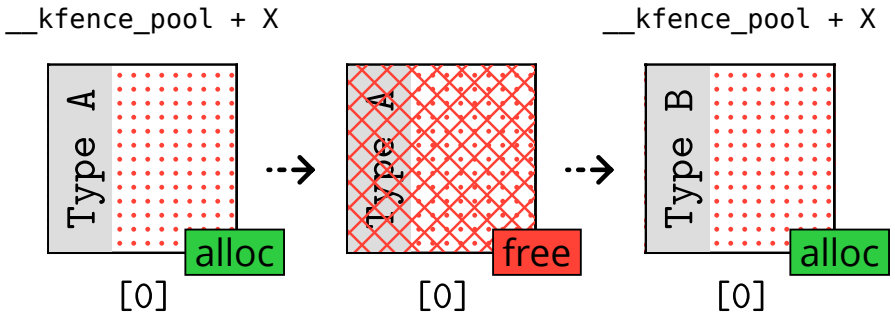
SLAB_VIRTUAL

> KFENCE does not honor SLAB_VIRTUAL hardened segregation guarantees

SLAB_VIRTUAL defends against cross-cache attacks by partitioning the VA space:



Cross-cache **use-after-free** exploitation can be **revived** under KFENCE



```
[139] vm_area_struct (?)
[140] skbuff_fclone_cache (?)
[141] skbuff_fclone_cache (?)
[142] skbuff_head_cache (?)
[143] vm_area_struct (?)
[144] vm_area_struct (?)
[145] skbuff_fclone_cache (?)
[146] vm_area_struct (?)
[147] vm_area_struct (?)
[148] vm_area_struct (?)
[149] vm_area_struct (?)
[150] skbuff_fclone_cache (?)
[151] vm_area_struct (?)
[152] <unused>
[153] vm_area_struct (?)
[154] vm_area_struct (?)
[155] kmalloc-rnd-12-192 (?)
[156] vm_area_struct (?)
[157] vm_area_struct (?)
[158] skbuff_fclone_cache (?)
[159] vm_area_struct (?)
[160] vm_area_struct (?)
[161] skbuff_fclone_cache (?)
[162] vm_area_struct (?)
[163] vm_area_struct (?)
[164] vm_area_struct (?)
[165] vm_area_struct (?)
[166] <unused>
[167] skbuff_fclone_cache (?)
[168] vm_area_struct (?)
[169] vm_area_struct (?)
[170] <unused>
[171] skbuff_fclone_cache (?)
[172] vm_area_struct (?)
[173] vm_area_struct (?)
[174] vm_area_struct (?)
[175] <unused>
[176] skbuff_fclone_cache (?)
[177] vm_area_struct (?)
[178] <unused>
[179] skbuff_fclone_cache (?)
[180] vm_area_struct (?)
[181] vm_area_struct (?)
[182] vm_area_struct (?)
[183] vm_area_struct (?)
[184] vm_area_struct (?)
[185] skbuff_head_cache (?)
[186] vm_area_struct (?)
[187] skbuff_head_cache (?)
[188] dyn-kmalloc-rnd-10-512 (?)
[189] maple_node (?)
[190] skbuff_fclone_cache (?)
[191] vm_area_struct (?)
[192] skbuff_fclone_cache (?)
[193] vm_area_struct (?)
[194] skbuff_fclone_cache (?)
[195] skbuff_fclone_cache (?)
[196] skbuff_fclone_cache (?)
[197] <unused>
[198] skbuff_head_cache (?)
[199] skbuff_fclone_cache (?)
[200] vm_area_struct (?)
[201] vm_area_struct (?)
```

```
while(1) {
  alloc_msg();
  alloc_cred();
}
```


[139] ... ed (?) -> sigqueue (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> kmalloc-rnd-04-96 (?) -> cred (?) -> vm_area_struct (?) -> cred (?) -> kmalloc-rnd-04-96 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> maple_node (?) -> cred (?)
[140] ... le_node (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> vm_area_struct (?) -> cred (?) -> pid (?) -> maple_node (?) -> cred (?) -> msg_msg-64 (?) -> sigqueue (?) -> cred (?) -> msg_msg-64 (?) -> dyn-kmalloc-rnd-06-8 (?)
[141] ... d (?) -> pid (?) -> cred (?) -> dyn-kmalloc-rnd-06-8 (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> kmalloc-rnd-04-96 (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> kmalloc-rnd-04-96 (?) -> cred (?) -> msg_msg-64 (?)
[142] ... pid (?) -> vm_area_struct (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> fs_cache (?) -> msg_msg-64 (?) -> cred (?) -> dyn-kmalloc-rnd-06-8 (?) -> vm_area_struct (?) -> msg_msg-64 (?) -> anon_vma (?)
[143] ... cred (?) -> cred (?) -> sigqueue (?) -> cred (?) -> cred (?) -> cred (?) -> sigqueue (?) -> msg_msg-64 (?) -> pid (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> kmalloc-rnd-04-96 (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?)
[144] ... ode (?) -> sigqueue (?) -> cred (?) -> cred (?) -> fs_cache (?) -> msg_msg-64 (?) -> cred (?) -> maple_node (?) -> vm_area_struct (?) -> cred (?) -> cred (?) -> sigqueue (?) -> kmalloc-rnd-04-96 (?) -> vm_area_struct (?) -> cred (?) -> msg_msg-64 (?)
[145] ... ed (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> dyn-kmalloc-rnd-06-8 (?) -> dyn-kmalloc-rnd-06-8 (?) -> dyn-kmalloc-rnd-06-8 (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> vma_lock (?) -> msg_msg-64 (?) -> msg_msg-64 (?)
[146] ... malloc-rnd-06-8 (?) -> msg_msg-64 (?) -> vm_area_struct (?) -> msg_msg-64 (?) -> dyn-kmalloc-rnd-06-8 (?) -> signal_cache (?) -> kmalloc-rnd-04-96 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> dyn-kmalloc-rnd-06-8 (?) -> cred (?) -> msg_msg-64 (?)
[147] ... (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> skbuff_fclone_cache (?) -> cred (?) -> cred (?) -> dyn-kmalloc-rnd-06-8 (?) -> cred (?) -> msg_msg-64 (?) -> cred (?)
[148] ... 64 (?) -> cred (?) -> anon_vma_chain (?) -> cred (?) -> vm_area_struct (?) -> cred (?) -> dyn-kmalloc-rnd-06-8 (?) -> cred (?) -> dyn-kmalloc-rnd-07-32 (?) -> dyn-kmalloc-rnd-07-32 (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> maple_node (?)
[149] ... de (?) -> dyn-kmalloc-rnd-06-8 (?) -> cred (?) -> cred (?) -> vm_area_struct (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> dyn-kmalloc-rnd-06-8 (?) -> msg_msg-64 (?) -> maple_node (?) -> cred (?) -> vm_area_struct (?) -> msg_msg-64 (?) -> cred (?)
[150] ... lloc-rnd-06-8 (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> maple_node (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> pid (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?)
[151] ...) -> vm_area_struct (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> maple_node (?) -> dyn-kmalloc-rnd-06-8 (?) -> cred (?) -> kmalloc-rnd-04-96 (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?)
[152] ... (?) -> cred (?) -> cred (?) -> vm_area_struct (?) -> cred (?) -> cred (?) -> dyn-kmalloc-rnd-06-8 (?) -> skbuff_fclone_cache (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> kmalloc-rnd-04-96 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?)
[153] ... ea_struct (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> dyn-kmalloc-rnd-06-8 (?) -> kmalloc-rnd-04-96 (?) -> cred (?) -> anon_vma (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?)
[154] ... (?) -> cred (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> skbuff_fclone_cache (?) -> vm_area_struct (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?)
[155] ... m_area_struct (?) -> cred (?) -> sigqueue (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> dyn-kmalloc-rnd-06-8 (?) -> cred (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> sighand_cache (?)
[156] ... de (?) -> maple_node (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> dyn-kmalloc-rnd-06-8 (?) -> maple_node (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> pid (?) -> kmalloc-rnd-04-96 (?) -> cred (?) -> cred (?) -> cred (?) -> cred (?) -> cred (?) -> cred (?)
[157] ...) -> dyn-kmalloc-rnd-06-8 (?) -> msg_msg-64 (?) -> cred (?) -> dyn-kmalloc-rnd-06-8 (?) -> dyn-kmalloc-rnd-07-32 (?) -> cred (?) -> msg_msg-64 (?) -> dyn-kmalloc-rnd-06-8 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> vm_area_struct (?)
[158] ... -rnd-06-8 (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> kmalloc-rnd-04-96 (?) -> msg_msg-64 (?) -> cred (?) -> vm_area_struct (?) -> vma_lock (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> cred (?)
[159] ... oc-rnd-04-96 (?) -> cred (?) -> msg_msg-64 (?) -> sigqueue (?) -> msg_msg-64 (?) -> vm_area_struct (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> cred (?) -> vm_area_struct (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?)
[160] ... -> dyn-kmalloc-rnd-06-8 (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> cred (?) -> vm_area_struct (?) -> msg_msg-64 (?) -> dyn-kmalloc-rnd-06-8 (?) -> dyn-kmalloc-rnd-06-8 (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?)
[161] ... ?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> mm_struct (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> pid (?) -> cred (?) -> cred (?) -> cred (?) -> vm_area_struct (?)
[162] ... uct (?) -> cred (?) -> maple_node (?) -> vm_area_struct (?) -> vm_area_struct (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> anon_vma (?) -> pid (?) -> msg_msg-64 (?) -> cred (?) -> fs_cache (?) -> cred (?) -> msg_msg-64 (?) -> cred (?)
[163] ... ?) -> sigqueue (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> cred (?) -> vm_area_struct (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> cred (?) -> cred (?) -> cred (?)
[164] ... loc-rnd-06-8 (?) -> cred (?) -> dyn-kmalloc-rnd-06-8 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> kmalloc-rnd-04-96 (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> skbuff_fclone_cache (?) -> vm_area_struct (?) -> cred (?) -> cred (?) -> msg_msg-64 (?)
[165] dyn-kmalloc-rnd-13-64 (?)
[166] ... malloc-rnd-04-96 (?) -> cred (?) -> sigqueue (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> dyn-kmalloc-rnd-06-8 (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?)
[167] ... (?) -> cred (?) -> cred (?) -> cred (?) -> cred (?) -> cred (?) -> vm_area_struct (?) -> cred (?) -> msg_msg-64 (?) -> sigqueue (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> vm_area_struct (?) -> msg_msg-64 (?)
[168] ... msg_msg-64 (?) -> cred (?) -> cred (?) -> mm_struct (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> vm_area_struct (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> dyn-kmalloc-rnd-06-8 (?) -> cred (?) -> msg_msg-64 (?) -> cred (?)
[169] ...) -> msg_msg-64 (?) -> kmalloc-rnd-04-96 (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> cred (?) -> dyn-kmalloc-rnd-06-8 (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> sigqueue (?)
[170] ... ?) -> vm_area_struct (?) -> sigqueue (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> cred (?)
[171] ... (?) -> vm_area_struct (?) -> msg_msg-64 (?) -> cred (?) -> sigqueue (?) -> pid (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> cred (?) -> cred (?) -> sigqueue (?) -> cred (?) -> dyn-kmalloc-rnd-06-8 (?) -> dyn-kmalloc-rnd-06-8 (?)
[172] ... 6 (?) -> cred (?) -> sigqueue (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> cred (?) -> vm_area_struct (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> sighand_cache (?)
[173] dyn-kmalloc-rnd-06-128 (?)
[174] ... g-64 (?) -> cred (?) -> vm_area_struct (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> dyn-kmalloc-rnd-06-8 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?)
[175] ... d (?) -> vm_area_struct (?) -> kmalloc-rnd-04-96 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> vm_area_struct (?) -> cred (?) -> kmalloc-rnd-04-96 (?) -> msg_msg-64 (?) -> cred (?) -> files_cache (?) -> cred (?) -> maple_node (?) -> cred (?) -> cred (?)
[176] ... -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> cred (?) -> kmalloc-rnd-04-96 (?) -> kmalloc-rnd-04-96 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> maple_node (?) -> pid (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> dyn-kmalloc-rnd-06-8 (?)
[177] ... msg_msg-64 (?) -> cred (?) -> sigqueue (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> vm_area_struct (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?)
[178] ... -> sigqueue (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> dyn-kmalloc-rnd-06-8 (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> signal_cache (?)
[179] ... struct (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?)
[180] ... 64 (?) -> msg_msg-64 (?) -> cred (?) -> vm_area_struct (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> maple_node (?) -> vm_area_struct (?) -> cred (?) -> cred (?) -> cred (?) -> dyn-kmalloc-rnd-06-8 (?) -> dyn-kmalloc-rnd-06-8 (?)
[181] ... _msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> kmalloc-rnd-04-96 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> vm_area_struct (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?)
[182] ... _struct (?) -> vm_area_struct (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> kmalloc-rnd-04-96 (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?)
[183] ... n-kmalloc-rnd-06-8 (?) -> msg_msg-64 (?) -> vm_area_struct (?) -> sigqueue (?) -> cred (?) -> maple_node (?) -> skbuff_fclone_cache (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> kmalloc-rnd-04-96 (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?)
[184] ... aple_node (?) -> cred (?) -> kmalloc-rnd-04-96 (?) -> cred (?) -> cred (?) -> vm_area_struct (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> dyn-kmalloc-rnd-06-8 (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> cred (?)
[185] ... > cred (?) -> sigqueue (?) -> msg_msg-64 (?) -> cred (?) -> sigqueue (?) -> vm_area_struct (?) -> msg_msg-64 (?) -> vma_lock (?) -> cred (?) -> msg_msg-64 (?) -> sigqueue (?) -> cred (?) -> cred (?) -> cred (?) -> cred (?) -> dyn-kmalloc-rnd-06-8 (?)
[186] ... 64 (?) -> cred (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> sigqueue (?) -> cred (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?)
[187] ... d (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> pid (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?)
[188] ... 4 (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> kmalloc-rnd-04-96 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> dyn-kmalloc-rnd-06-8 (?)
[189] ... (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> vm_area_struct (?) -> vm_area_struct (?) -> vm_area_struct (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?)
[190] ... red (?) -> cred (?) -> kmalloc-rnd-04-96 (?) -> cred (?) -> sigqueue (?) -> cred (?) -> maple_node (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> dyn-kmalloc-rnd-06-8 (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?)
[191] ... ?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> dyn-kmalloc-rnd-06-8 (?) -> msg_msg-64 (?) -> msg_ms
[192] ... > cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> kmalloc-rnd-04-96 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> c
[193] ... (?) -> msg_msg-64 (?) -> dyn-kmalloc-rnd-06-8 (?) -> dyn-kmalloc-rnd-06-8 (?) -> cred (?) -> maple_node (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> dyn-km
[194] ... _msg-64 (?) -> kmalloc-rnd-04-96 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> dyn-kmalloc-rnd-06-8 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> cred (?)
[195] ... -04-96 (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> dyn-kmalloc-rnd-06-8 (?) -> cred (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> anon_vma (?) -> cred (?) -> msg_ms
[196] ... -64 (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> vm_area_struct (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> kmalloc-rnd-04-96 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> anon_vma (?)
[197] ... d (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> sigqueue (?) -> msg_msg-64 (?) -> cred (?) -> kmalloc-rnd-04-96 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> cr
[198] ... a_struct (?) -> cred (?) -> cred (?) -> kmalloc-rnd-04-96 (?) -> vm_area_struct (?) -> cred (?) -> dyn-kmalloc-rnd-06-8 (?) -> msg_msg-64 (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> map
[199] ... msg_msg-64 (?) -> cred (?) -> cred (?) -> sigqueue (?) -> sigqueue (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> kmalloc-rnd-04-96 (?) -> cred (
[200] ... ed (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> cred (?) -> cred (?) -> cred (?) -> cred (?) -> vm_are
[201] ... 8 (?) -> cred (?) -> cred (?) -> cred (?) -> msg_msg-64 (?) -> sigqueue (?) -> cred (?) -> msg_msg-64 (?) -> msg_msg-64 (?) -> vm_area_struct (?) -> cred (?) -> kmalloc-rnd-04-96 (?) -> cred (?) -

1 while(1) {
2 alloc_msg();
3 alloc_cred();
4 }

KASAN Instrumentation

KASAN Instrumentation

> Objects served by KFENCE are not instrumented with KASAN

KASAN Instrumentation

> Objects served by KFENCE are not instrumented with KASAN

Most importantly, not under **`KASAN_HW_TAGS`** (MTE) production mode

KASAN Instrumentation

> Objects served by KFENCE are not instrumented with KASAN

Most importantly, not under **KASAN_HW_TAGS** (MTE) production mode

```
1  void * __kasan_slab_alloc(..., void *object)
2  {
3    // ...
4    if (is_kfence_address(object))
5      return (void *)object;
6
7    tag = assign_tag(cache, object, false);
8    tagged_object = set_tag(object, tag);
9    // ...
10   return tagged_object;
11 }
```

KASAN Instrumentation

> Objects served by KFENCE are not instrumented with KASAN

Most importantly, not under **KASAN_HW_TAGS** (MTE) production mode

```
1  void * __kasan_slab_alloc(..., void *object)
2  {
3    // ...
4    if (is_kfence_address(object))
5      return (void *)object;
6
7    tag = assign_tag(cache, object, false);
8    tagged_object = set_tag(object, tag);
9    // ...
10   return tagged_object;
11 }
```

Let’s do another experiment

Kernel

```
1  switch (cmd) {
2    case TEST_IOCTL_ALLOC:
3      // ...
4      new_buf = kmalloc(alloc_req.size, GFP_KERNEL);
5      // ...
6      if (copy_to_user((void __user *)arg, ..., ...))
7        BUG_ON(1);
8
9      return 0;
```

Kernel

```
switch (cmd) {
  case TEST_IOCTL_ALLOC:
    // ...
    new_buf = kmalloc(alloc_req.size, GFP_KERNEL);
    // ...
    if (copy_to_user((void __user *)arg, ..., ...))
      BUG_ON(1);

    return 0;
```

+

Kernel

```
switch (cmd) {
  case TEST_IOCTL_ALLOC:
    // ...
    new_buf = kmalloc(alloc_req.size, GFP_KERNEL);
    // ...
    if (copy_to_user((void __user *)arg, ..., ...))
      BUG_ON(1);

    return 0;
```

+

User

```
while(1) {
  printf(..., alloc_kmalloc(...));
  usleep(1000); //1ms
}
```

Kernel

```
switch (cmd) {
  case TEST_IOCTL_ALLOC:
    // ...
    new_buf = kmalloc(alloc_req.size, GFP_KERNEL);
    // ...
    if (copy_to_user((void __user *)arg, ..., ...))
      BUG_ON(1);

    return 0;
```

+



User

```
while(1) {
  printf(..., alloc_kmalloc(...));
  usleep(1000); //1ms
}
```

```
# setprop arm64.memtag.bootctl memtag,memtag-kernel
```

```
kmalloc(..., GFP_KERNEL) -> 0xf8ffff8928695700 (slab  ) [MTE]
```

```
kmalloc(..., GFP_KERNEL) -> 0xf8ffff8928695700 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf0ffff88819e0040 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfdffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf4ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfaffff804e5eea40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf9ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf2ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf7ffff89179fcb40 (slab  ) [MTE]
```

```
kmalloc(..., GFP_KERNEL) -> 0xf8ffff8928695700 (slab  ) [MTE]

kmalloc(..., GFP_KERNEL) -> 0xf0ffff88819e0040 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfdffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf4ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfaffff804e5eea40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf9ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf2ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf7ffff89179fcb40 (slab  ) [MTE]

kmalloc(..., GFP_KERNEL) -> 0xffffff896a381fc0 (kfence) [NO MTE]
```

```
kmalloc(..., GFP_KERNEL) -> 0xf8ffff8928695700 (slab  ) [MTE]

kmalloc(..., GFP_KERNEL) -> 0xf0ffff88819e0040 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfdffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf4ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfaffff804e5eea40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf9ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf2ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf7ffff89179fcb40 (slab  ) [MTE]

kmalloc(..., GFP_KERNEL) -> 0xffffff896a381fc0 (kfence) [NO MTE]

kmalloc(..., GFP_KERNEL) -> 0xfbffff804e5eec80 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf7ffff8913860200 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf2ffff8913860200 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfdffff8913860200 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfcffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf0ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf8ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf9ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf0ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf9ffff89179fcb40 (slab  ) [MTE]
[...]
kmalloc(..., GFP_KERNEL) -> 0xf1ffff8926faf800 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfcffff8860329700 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfcffff8926faf800 (slab  ) [MTE]
```

```
kmalloc(..., GFP_KERNEL) -> 0xf8ffff8928695700 (slab  ) [MTE]

kmalloc(..., GFP_KERNEL) -> 0xf0ffff88819e0040 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfdffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf4ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfaffff804e5eea40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf9ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf2ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf7ffff89179fcb40 (slab  ) [MTE]

kmalloc(..., GFP_KERNEL) -> 0xffffff896a381fc0 (kfence) [NO MTE]

kmalloc(..., GFP_KERNEL) -> 0xfbffff804e5eec80 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf7ffff8913860200 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf2ffff8913860200 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfdffff8913860200 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfcffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf0ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf8ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf9ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf0ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf9ffff89179fcb40 (slab  ) [MTE]
[...]
kmalloc(..., GFP_KERNEL) -> 0xf1ffff8926faf800 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfcffff8860329700 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfcffff8926faf800 (slab  ) [MTE]

kmalloc(..., GFP_KERNEL) -> 0xffffff896a353fc0 (kfence) [NO MTE]
```

```
kmalloc(..., GFP_KERNEL) -> 0xf8ffff8928695700 (slab  ) [MTE]

kmalloc(..., GFP_KERNEL) -> 0xf0ffff88819e0040 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfdffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf4ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfaffff804e5eea40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf9ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf2ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf7ffff89179fcb40 (slab  ) [MTE]

kmalloc(..., GFP_KERNEL) -> 0xffffff896a381fc0 (kfence) [NO MTE]

kmalloc(..., GFP_KERNEL) -> 0xfbffff804e5eec80 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf7ffff8913860200 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf2ffff8913860200 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfdffff8913860200 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfcffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf0ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf8ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf9ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf0ffff89179fcb40 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf9ffff89179fcb40 (slab  ) [MTE]
[...]
kmalloc(..., GFP_KERNEL) -> 0xf1ffff8926faf800 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfcffff8860329700 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfcffff8926faf800 (slab  ) [MTE]

kmalloc(..., GFP_KERNEL) -> 0xffffff896a353fc0 (kfence) [NO MTE]

kmalloc(..., GFP_KERNEL) -> 0xf4ffff8926faf800 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf2ffff8860329700 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf0ffff8860329700 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xf0ffff8860329700 (slab  ) [MTE]
kmalloc(..., GFP_KERNEL) -> 0xfaffff89277ebc00 (slab  ) [MTE]
```

Downstream Hardening

Downstream Hardening

> KFENCE may weaken downstream hardening in unexpected ways

Downstream Hardening

> KFENCE may weaken downstream hardening in unexpected ways

Objects are not always backed by the same type of memory

Downstream Hardening

> KFENCE may weaken downstream hardening in unexpected ways

Objects are not always backed by the same type of memory

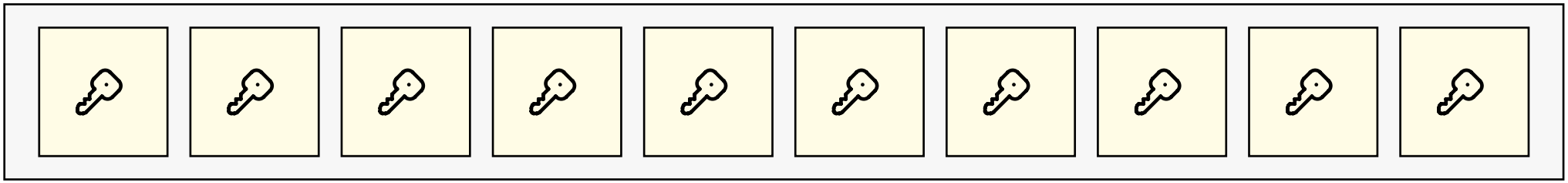
Hypervisors such as **Samsung RKP** mark sensitive slab pages as **read-only**:

Downstream Hardening

> KFENCE may weaken downstream hardening in unexpected ways

Objects are not always backed by the same type of memory

Hypervisors such as **Samsung RKP** mark sensitive slab pages as **read-only**:

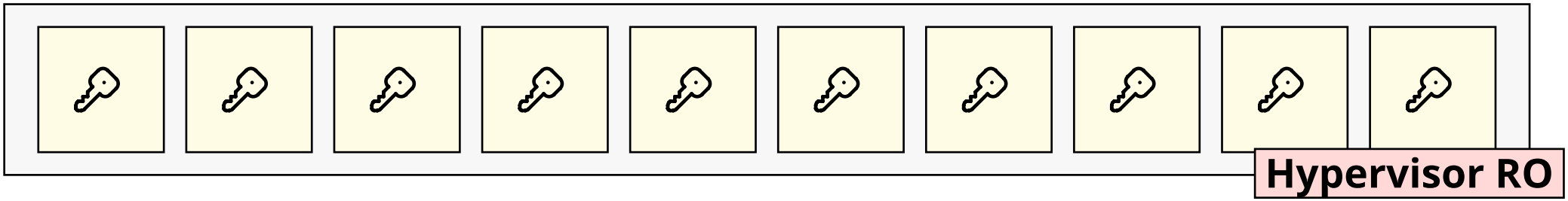


Downstream Hardening

> KFENCE may weaken downstream hardening in unexpected ways

Objects are not always backed by the same type of memory

Hypervisors such as **Samsung RKP** mark sensitive slab pages as **read-only**:

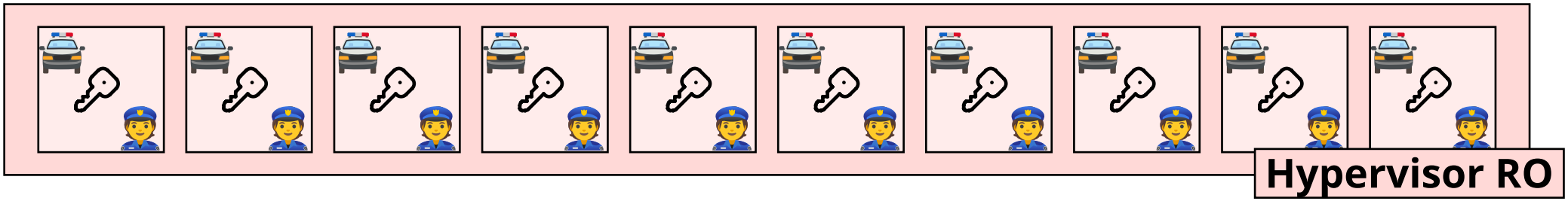


Downstream Hardening

> KFENCE may weaken downstream hardening in unexpected ways

Objects are not always backed by the same type of memory

Hypervisors such as **Samsung RKP** mark sensitive slab pages as **read-only**:

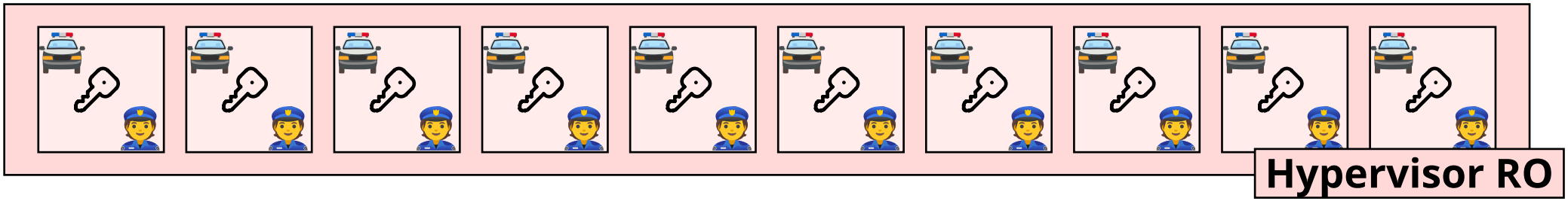


Downstream Hardening

> KFENCE may weaken downstream hardening in unexpected ways

Objects are not always backed by the same type of memory

Hypervisors such as **Samsung RKP** mark sensitive slab pages as **read-only**:



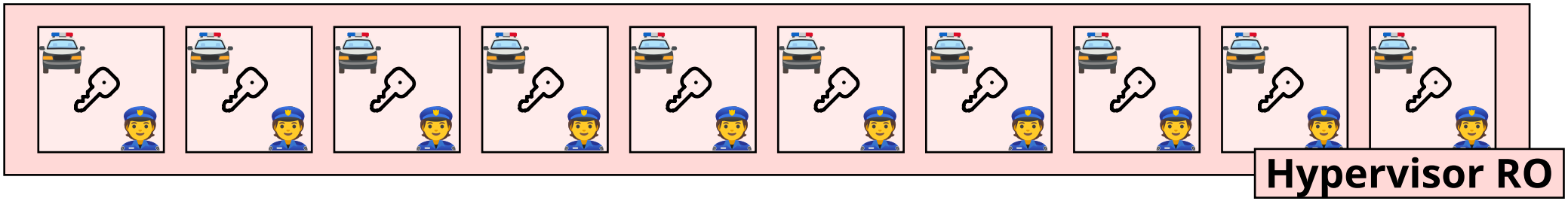
If you harden certain caches, you may need to **avoid KFENCE sampling**:

Downstream Hardening

> KFENCE may weaken downstream hardening in unexpected ways

Objects are not always backed by the same type of memory

Hypervisors such as **Samsung RKP** mark sensitive slab pages as **read-only**:



If you harden certain caches, you may need to **avoid KFENCE sampling**:

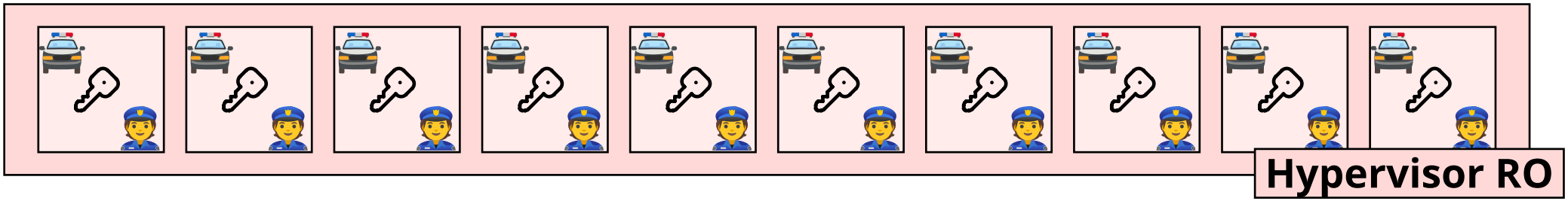
```
/sys/kernel/slab/cred/skip_kfence
```

Downstream Hardening

> KFENCE may weaken downstream hardening in unexpected ways

Objects are not always backed by the same type of memory

Hypervisors such as **Samsung RKP** mark sensitive slab pages as **read-only**:



If you harden certain caches, you may need to **avoid KFENCE sampling**:

```
/sys/kernel/slab/cred/skip_kfence
```

Other protections may have different quirks

Miscellaneous

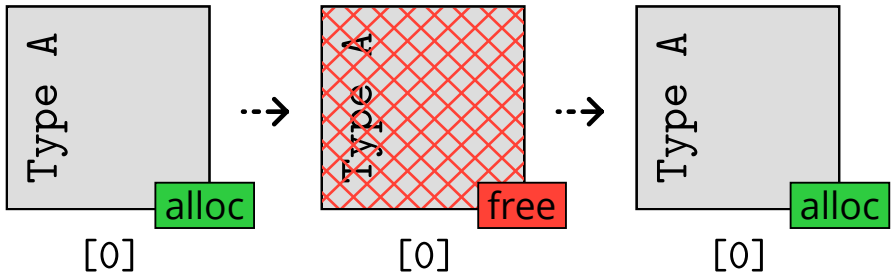
Miscellaneous

> Natural miscellaneous behavior that may benefit an attacker

Miscellaneous

> Natural miscellaneous behavior that may benefit an attacker

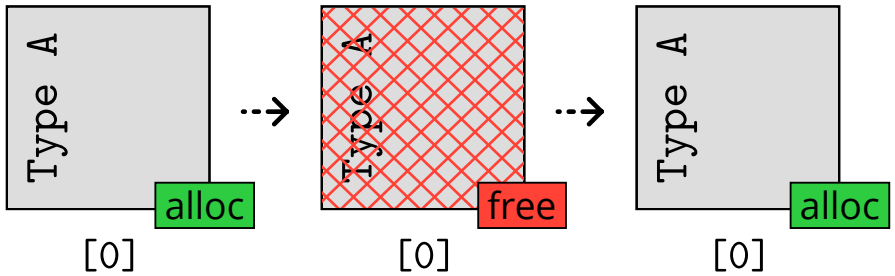
No 4K Object Randomization



Miscellaneous

> Natural miscellaneous behavior that may benefit an attacker

No 4K Object Randomization



Non-fatal Error Handling

Default & common mode is log-only

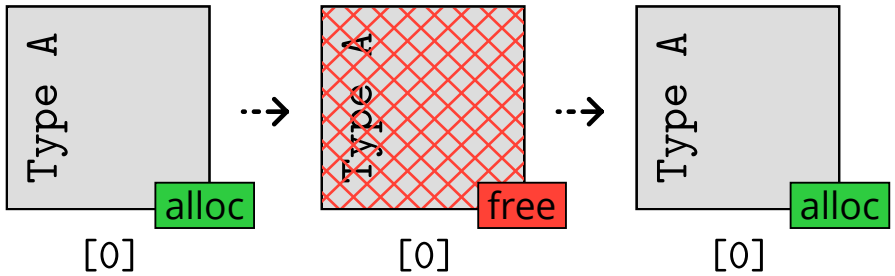
↳ Retry until privilege escalation

↳ Delete all logs

Miscellaneous

> Natural miscellaneous behavior that may benefit an attacker

No 4K Object Randomization



Deterministic Poison

```
gef> p (__kfence_pool + 2*4096)
$7 = 0xffff88813b402000 "\210<\230"
gef> hexdump -f qword 0xffff88813b402000 4096
```

```
0xffff88813b4020f0:    0x0000000000000000 0x0000000000000000
0xffff88813b402100:    0x0000000000000008 0xffffffff83  9b80
0xffff88813b402110:    0x0000000000000000 0x0000000000000000
0xffff88813b402120:    0x0000000000000000 0x0000000000000000
0xffff88813b402130:    0x0000000000000000 0x0000000000000000
0xffff88813b402140:    0xadacafaea9a8abaa 0xadacafaea9a8abaa
0xffff88813b402150:    0xadacafaea9a8abaa 0xadacafaea9a8abaa
0xffff88813b402160:    0xadacafaea9a8abaa 0xadacafaea9a8abaa
0xffff88813b402170:    0xadacafaea9a8abaa 0xadacafaea9a8abaa
0xffff88813b402180:    0xadacafaea9a8abaa 0xadacafaea9a8abaa
0xffff88813b402190:    0xadacafaea9a8abaa 0xadacafaea9a8abaa
0xffff88813b4021a0:    0xadacafaea9a8abaa 0xadacafaea9a8abaa
0xffff88813b4021b0:    0xadacafaea9a8abaa 0xadacafaea9a8abaa
0xffff88813b4021c0:    0xadacafaea9a8abaa 0xadacafaea9a8abaa
0xffff88813b4021d0:    0xadacafaea9a8abaa 0xadacafaea9a8abaa
0xffff88813b4021e0:    0xadacafaea9a8abaa 0xadacafaea9a8abaa
0xffff88813b4021f0:    0xadacafaea9a8abaa 0xadacafaea9a8abaa
0xffff88813b402200:    0xadacafaea9a8abaa 0xadacafaea9a8abaa
0xffff88813b402210:    0xadacafaea9a8abaa 0xadacafaea9a8abaa
0xffff88813b402220:    0xadacafaea9a8abaa 0xadacafaea9a8abaa
0xffff88813b402230:    0xadacafaea9a8abaa 0xadacafaea9a8abaa
0xffff88813b402240:    0xadacafaea9a8abaa 0xadacafaea9a8abaa
0xffff88813b402250:    0xadacafaea9a8abaa 0xadacafaea9a8abaa
0xffff88813b402260:    0xadacafaea9a8abaa 0xadacafaea9a8abaa
```

Non-fatal Error Handling

Default & common mode is log-only

↳ Retry until privilege escalation

↳ Delete all logs

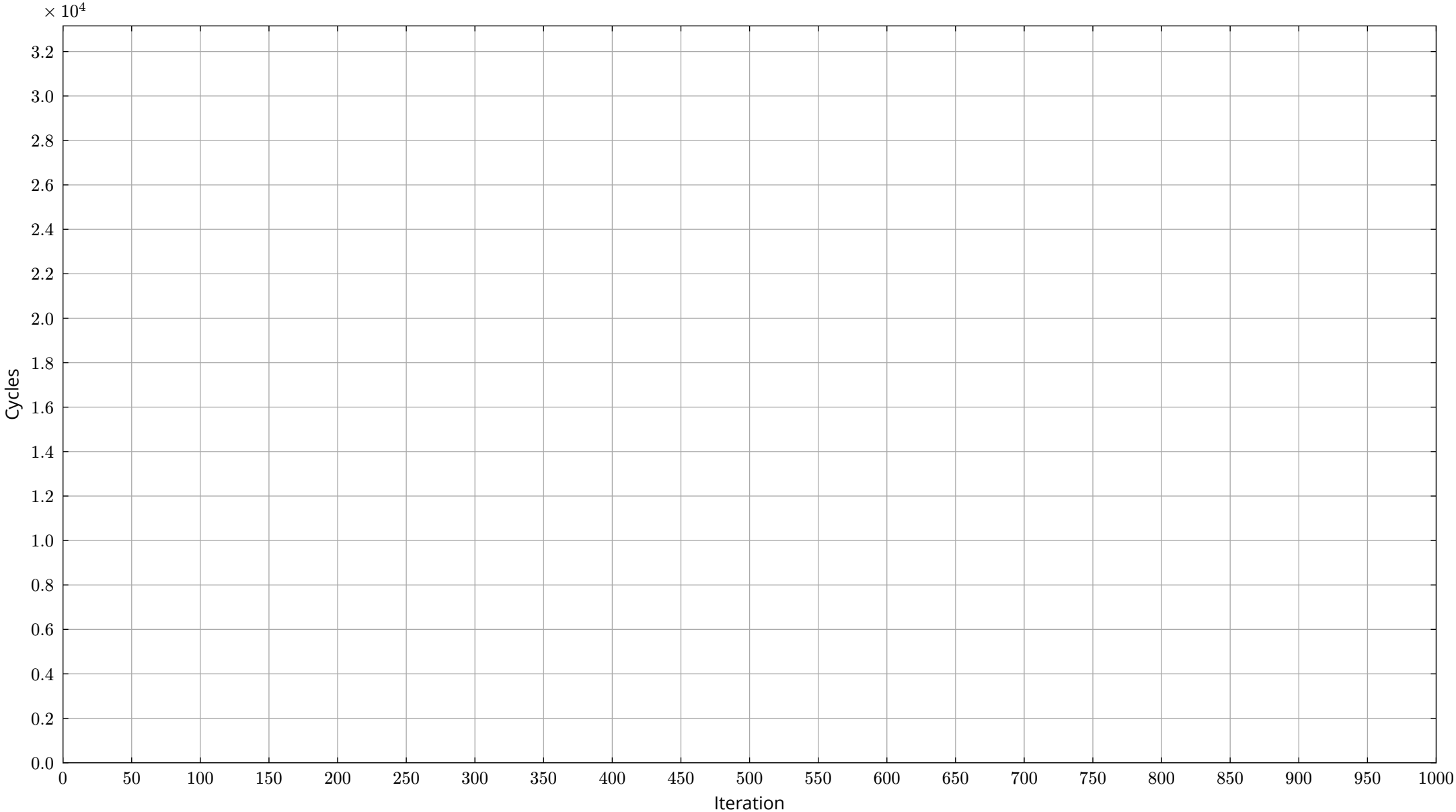
Fence2Pwn

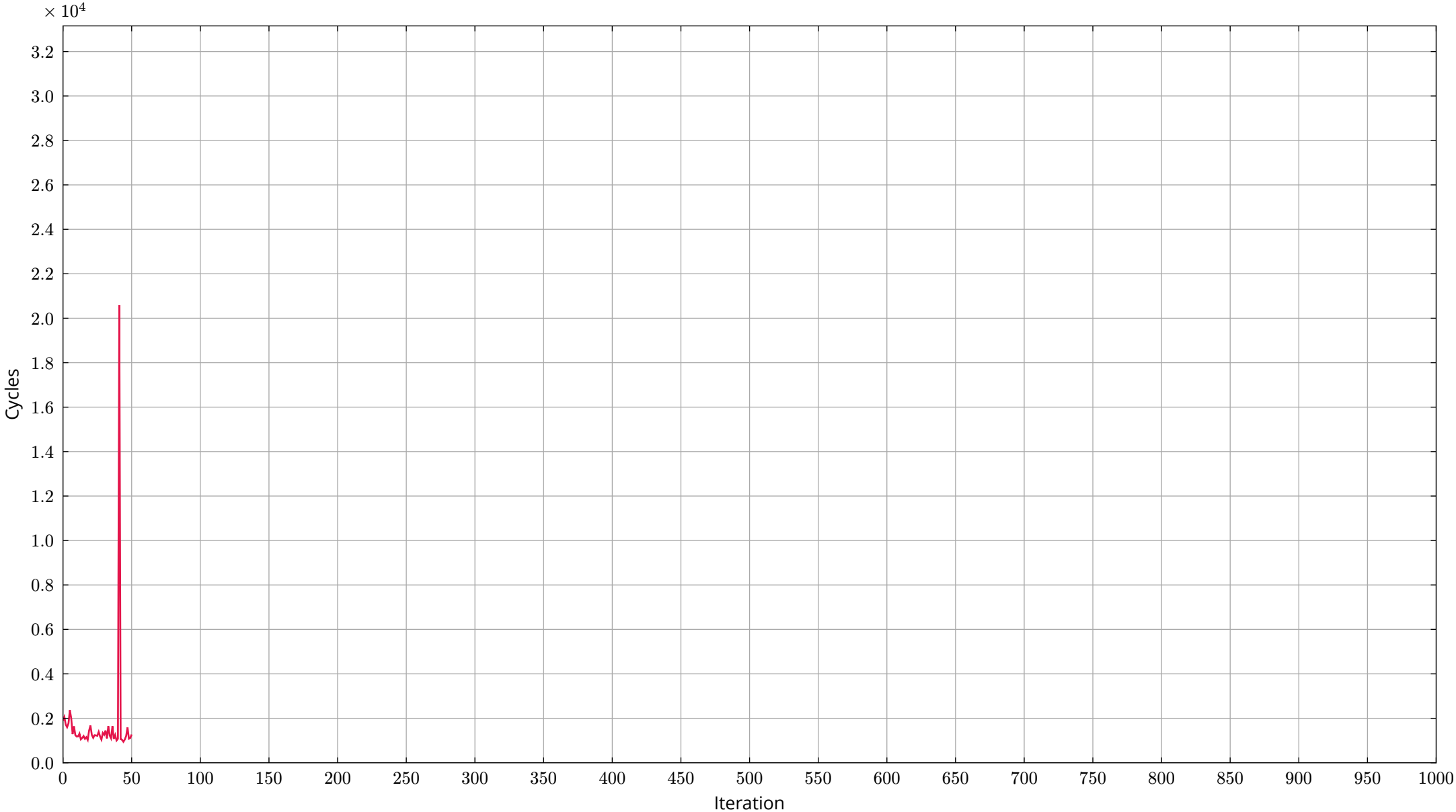


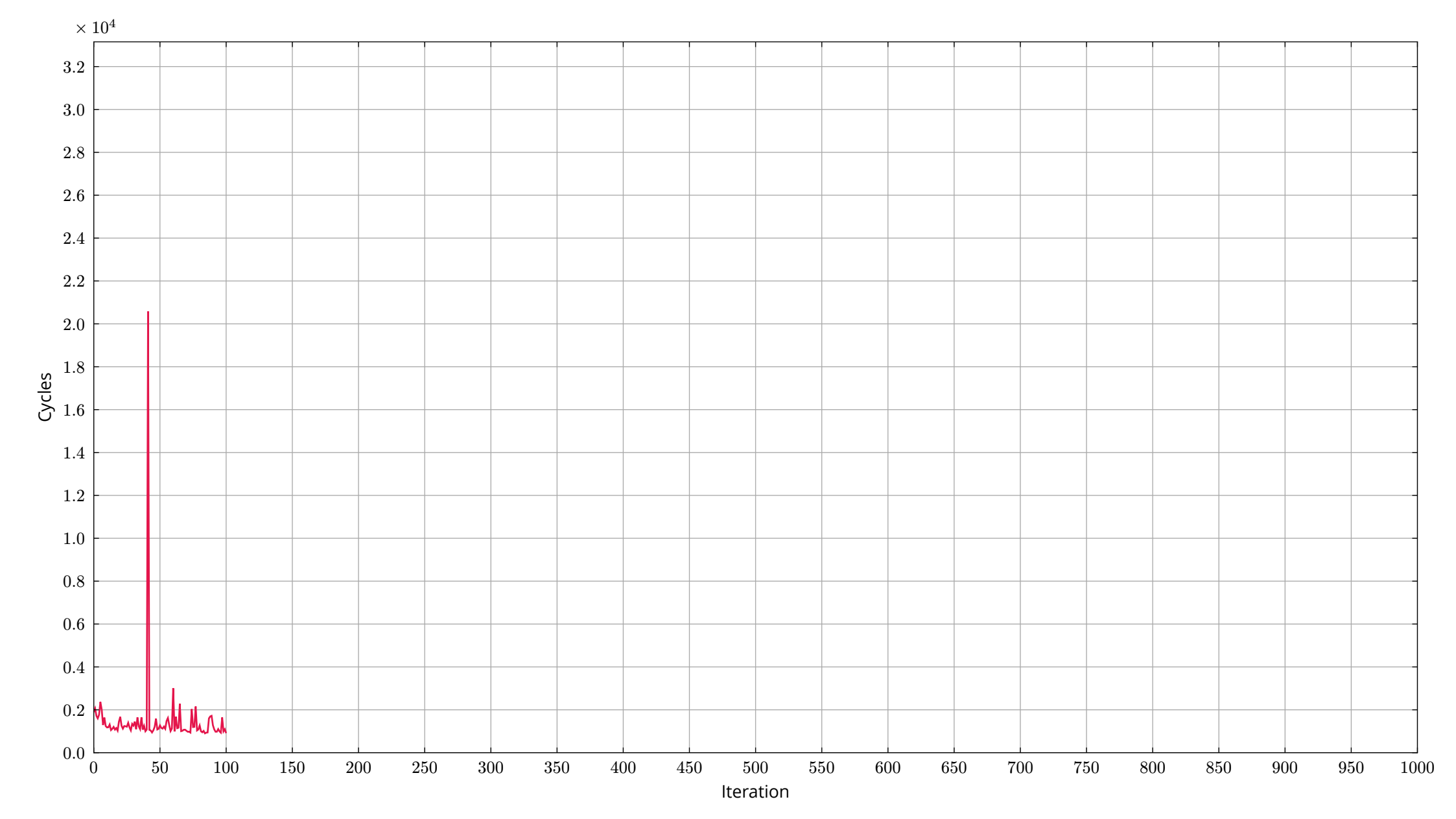
Experiment again!

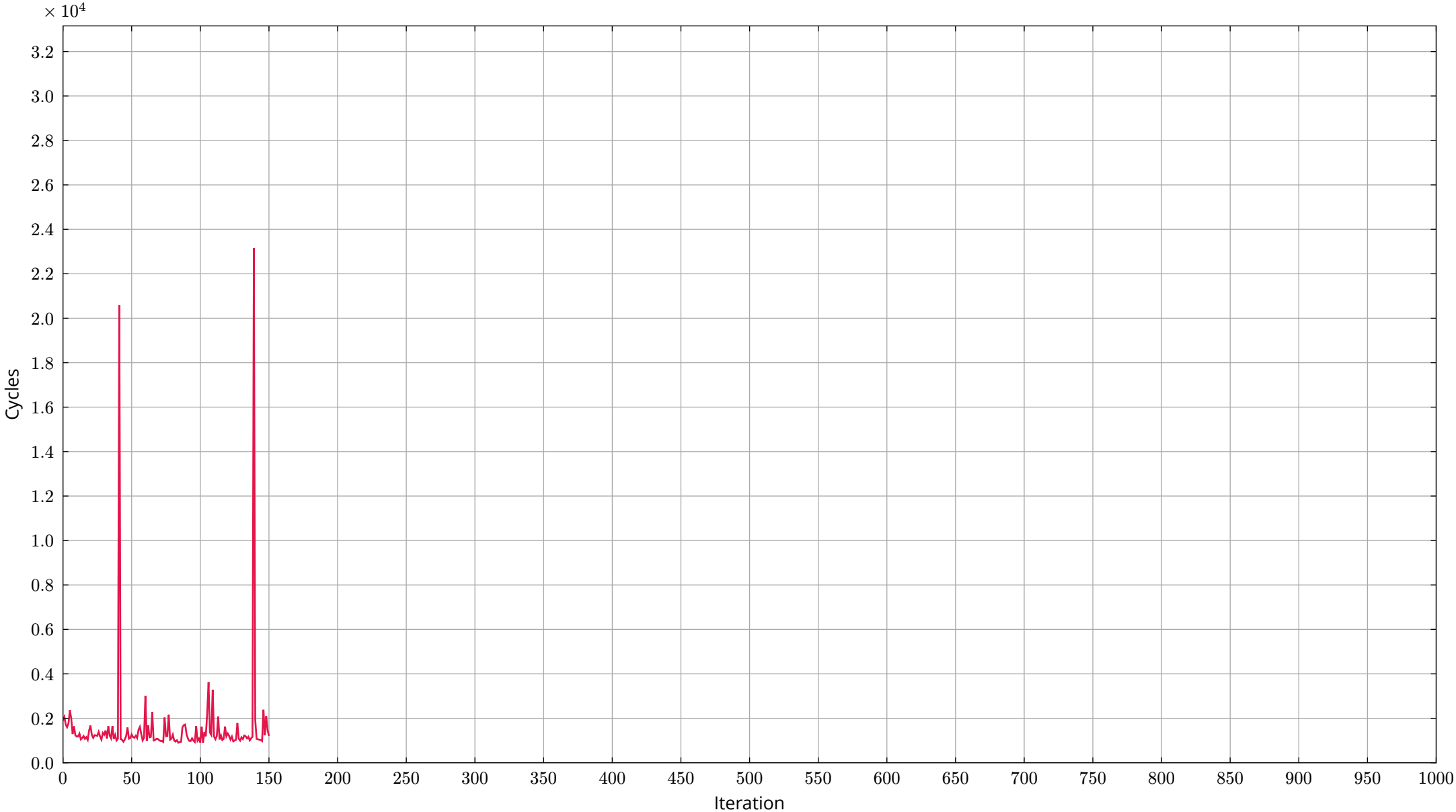
User

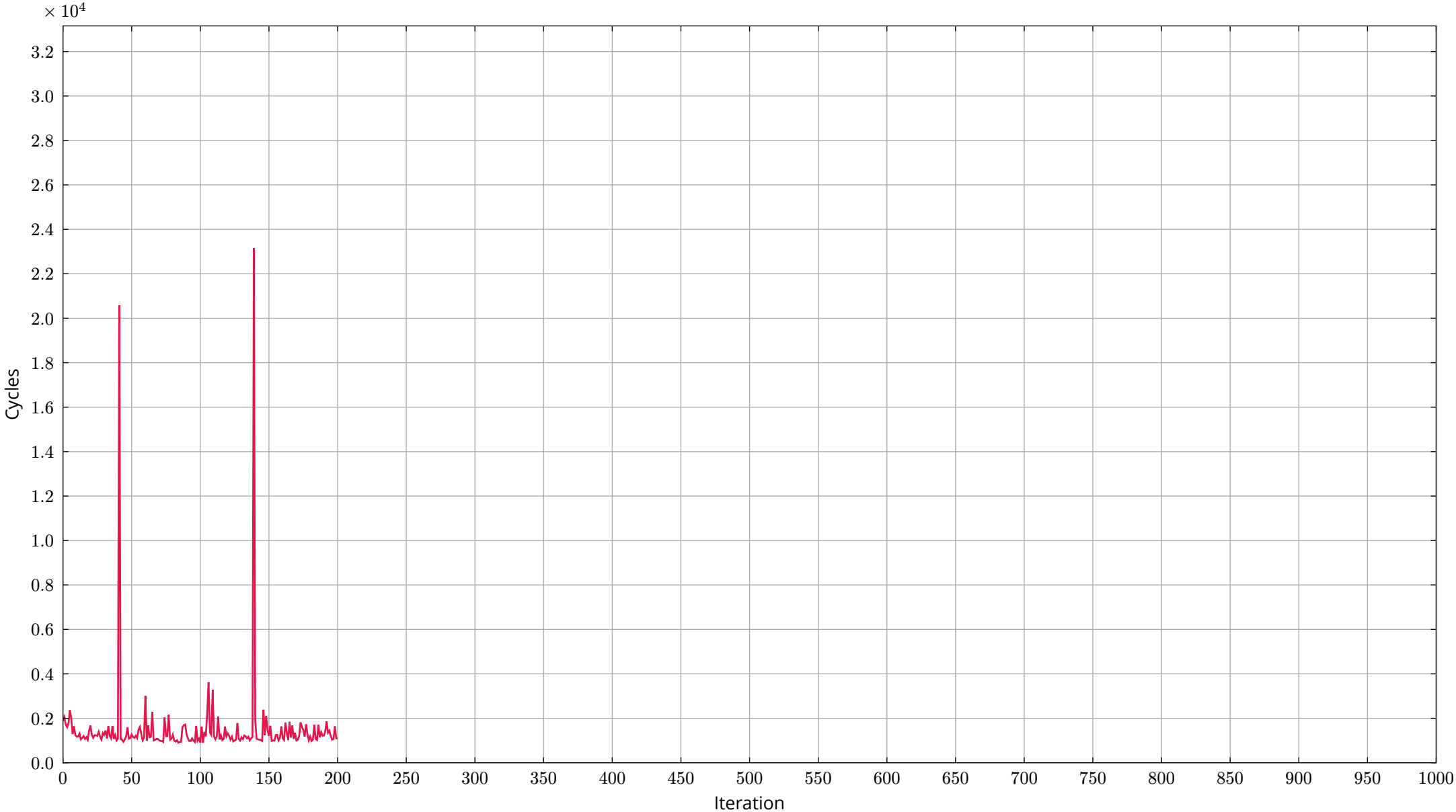
```c
pin_cpu(0);
while(1) {
  a = rdtsc();
  setgid(gid);
  b = rdtsc();
  printf(..., b - a);
  nanosleep(&(struct timespec){.tv_nsec = 1000000}, NULL); // 1ms
}
```

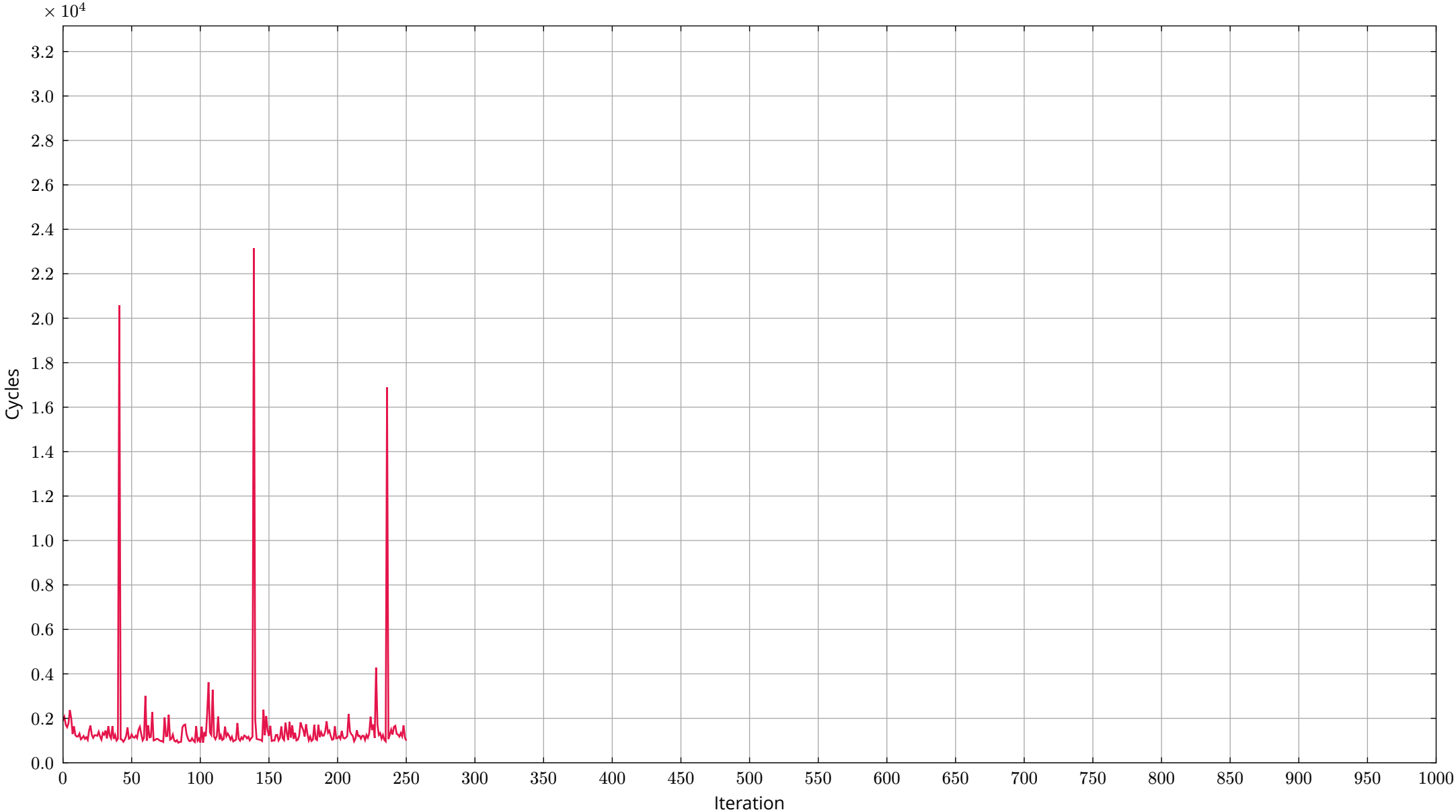

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration

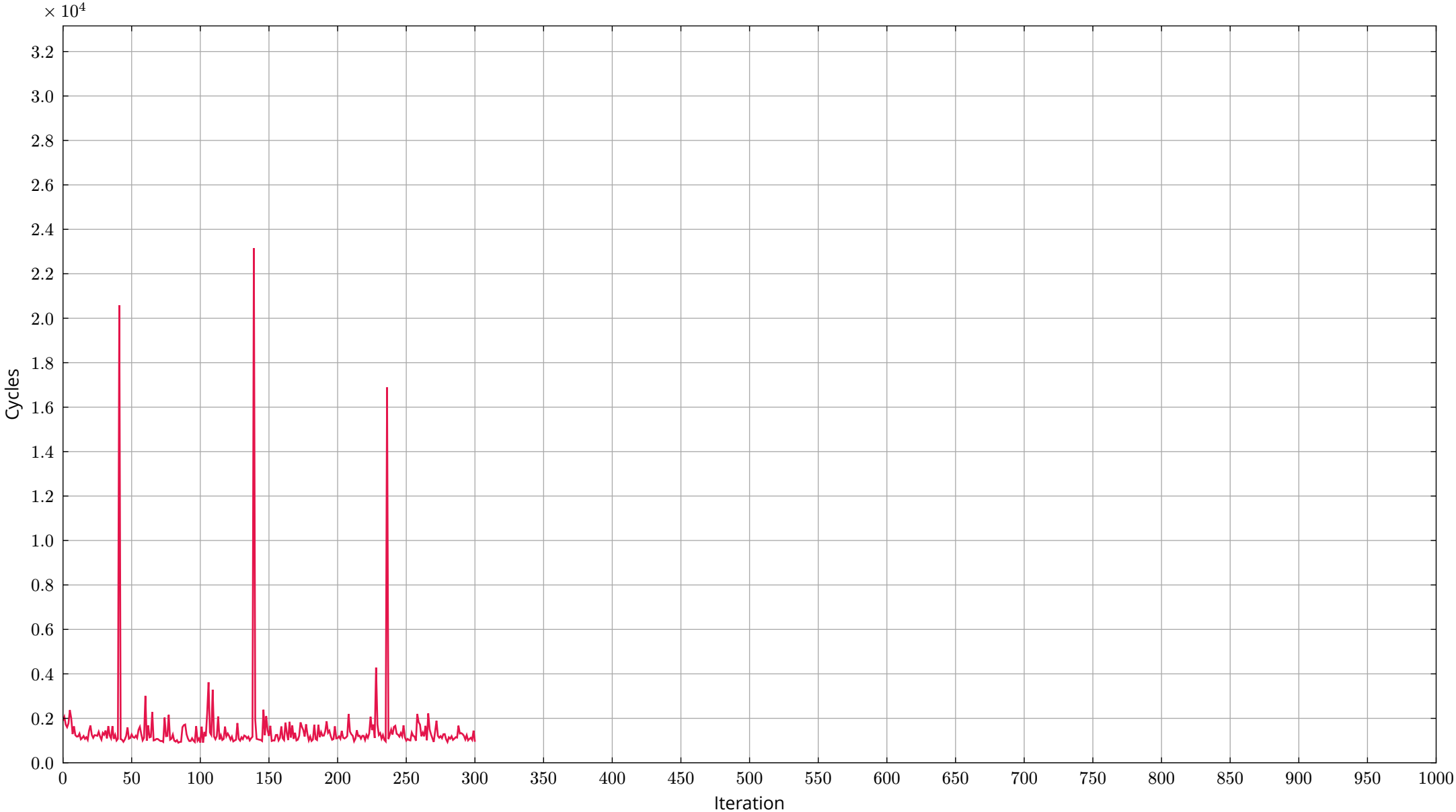

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration

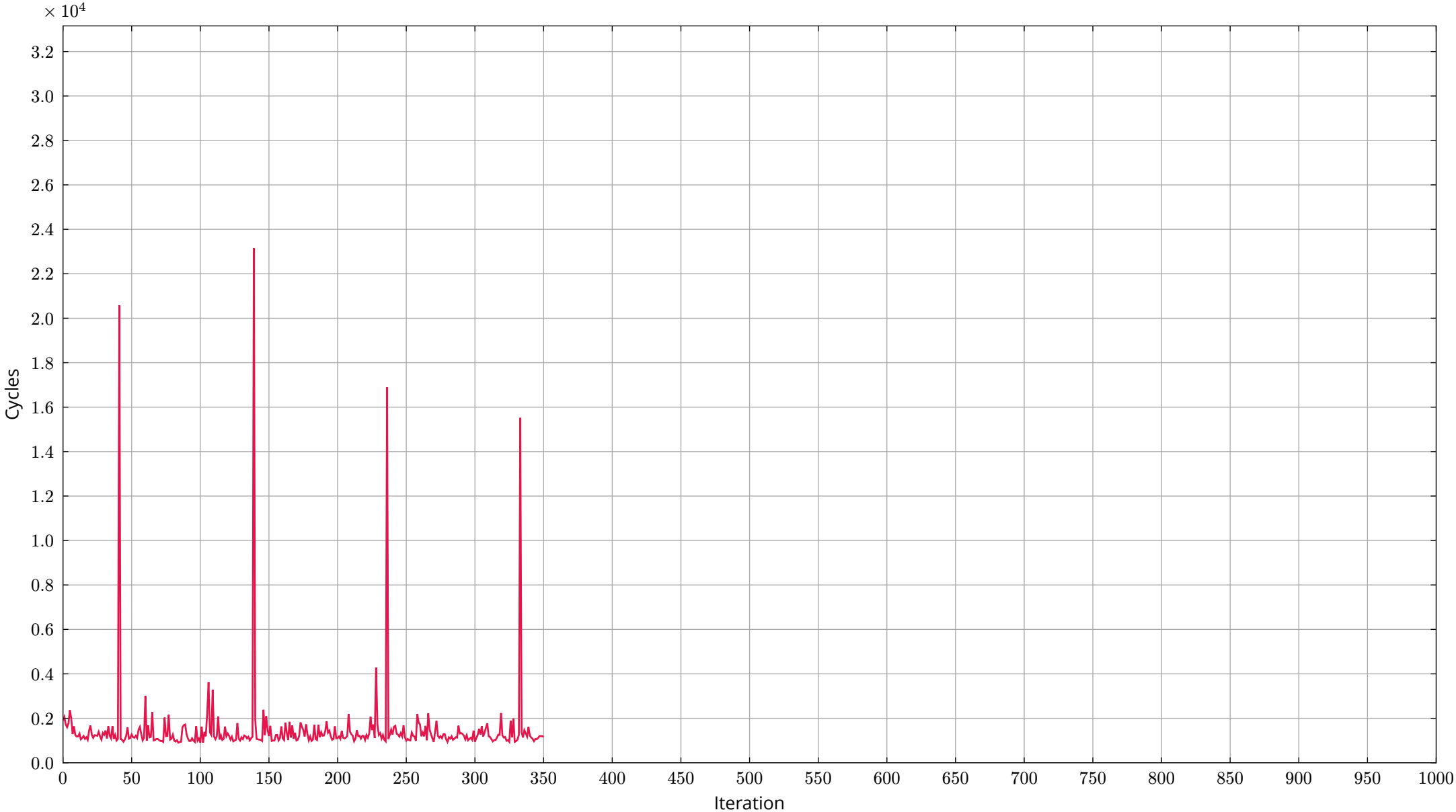

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration

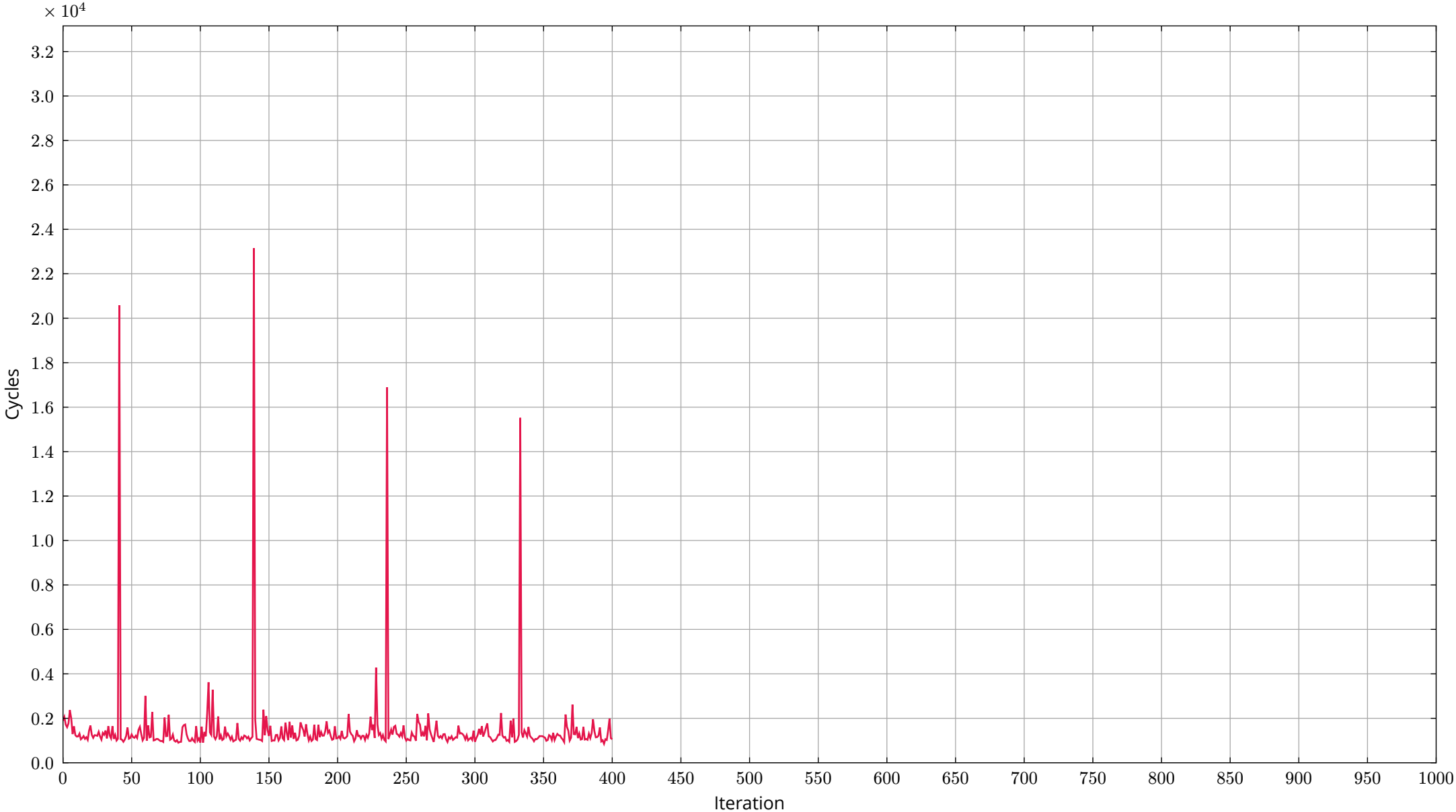

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration

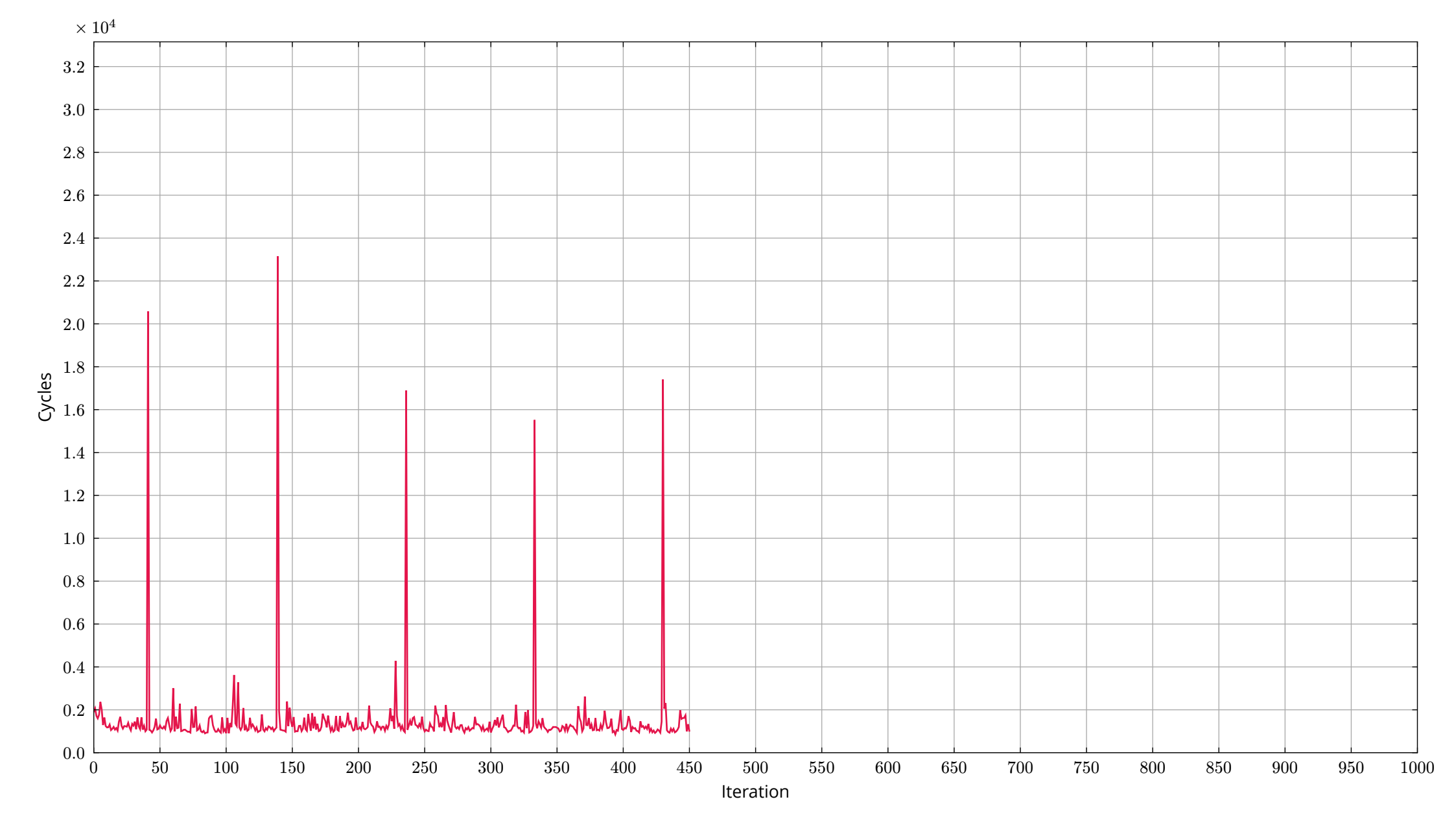

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration

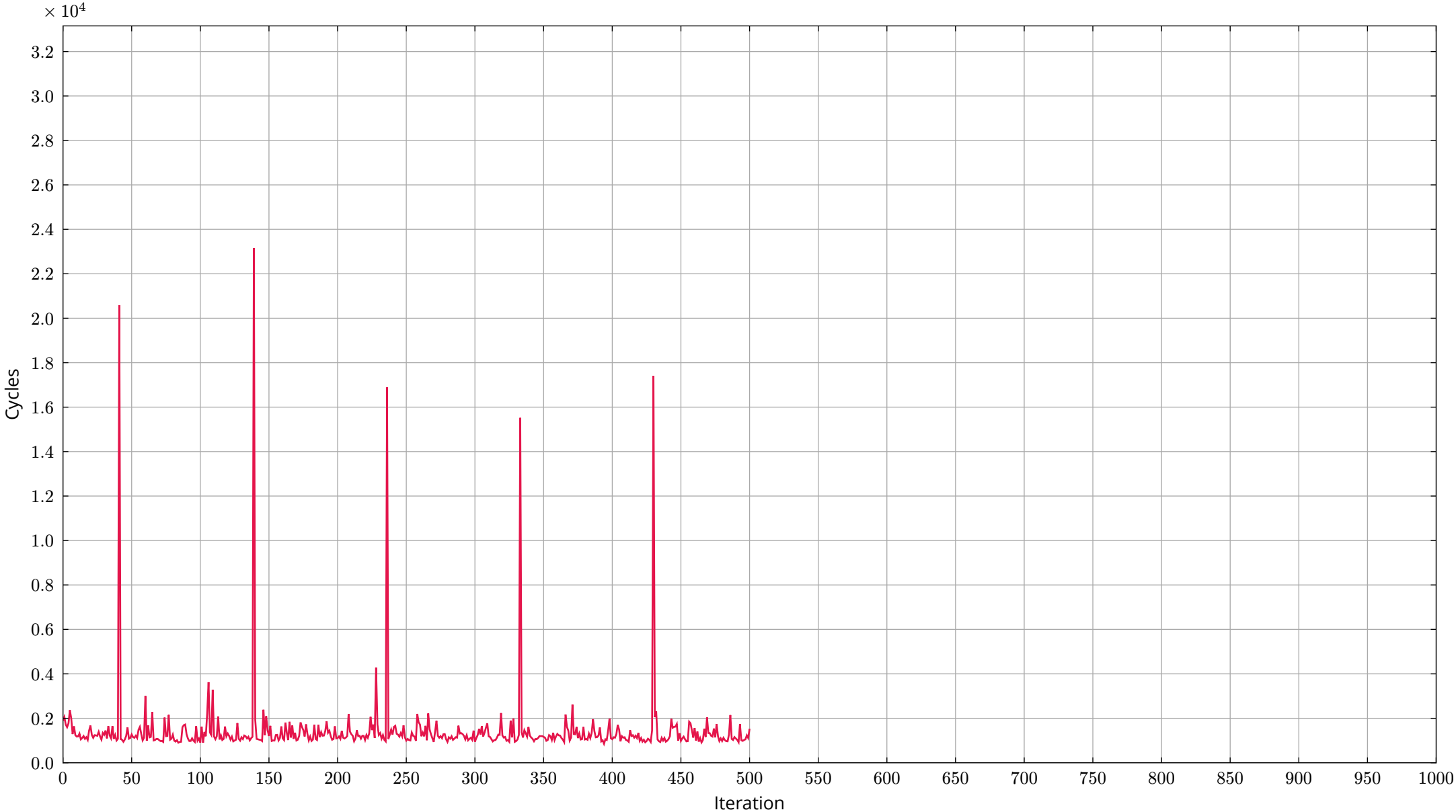

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration

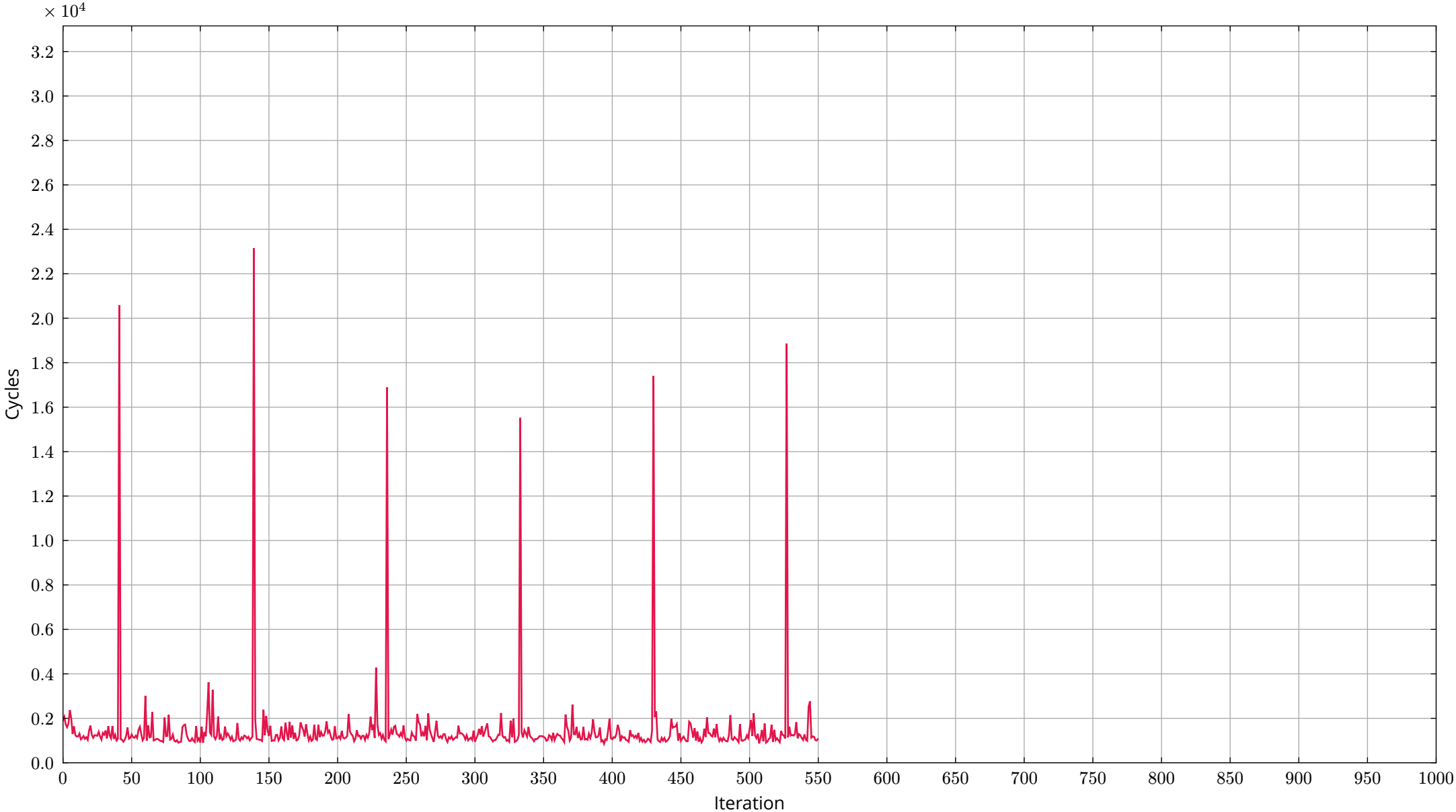

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration

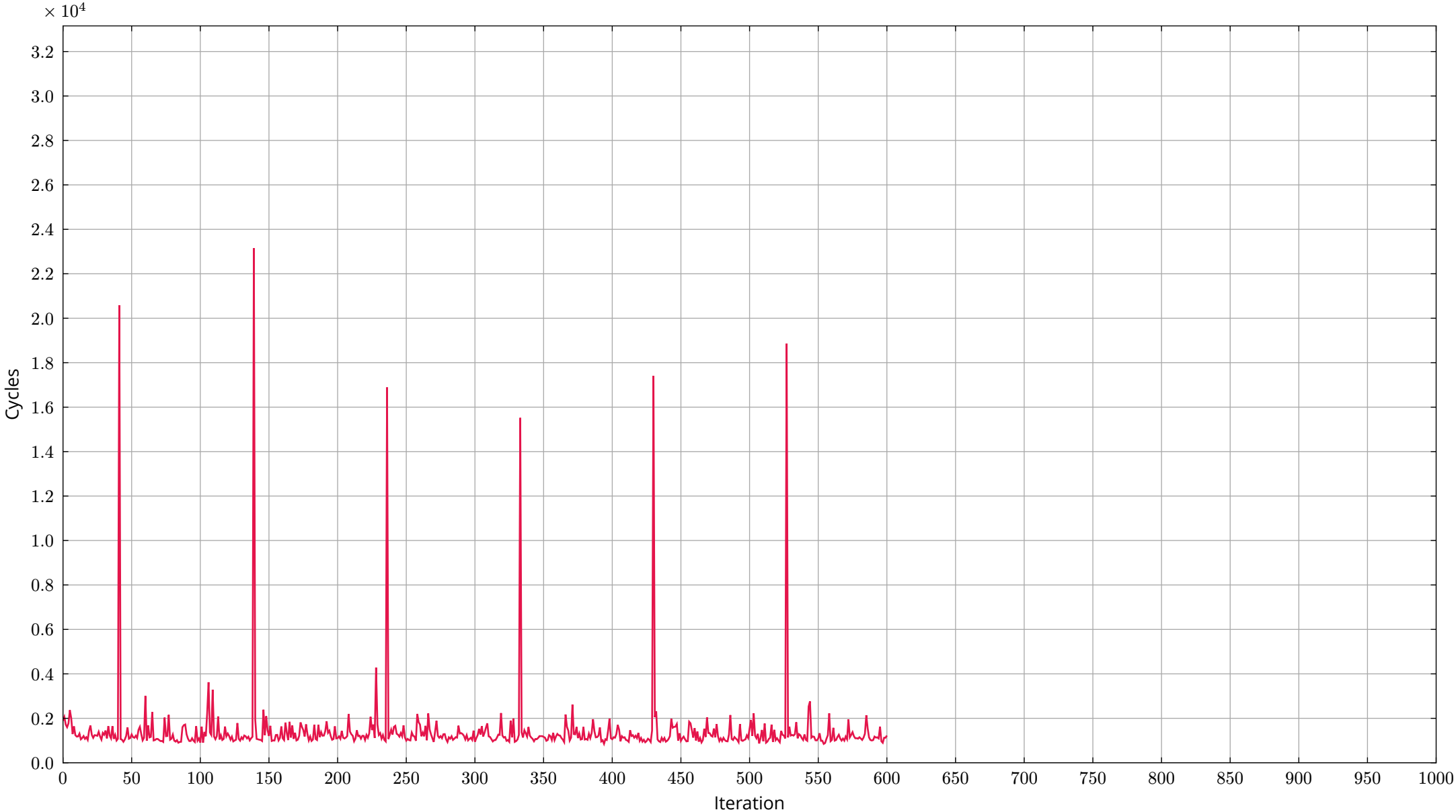

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration

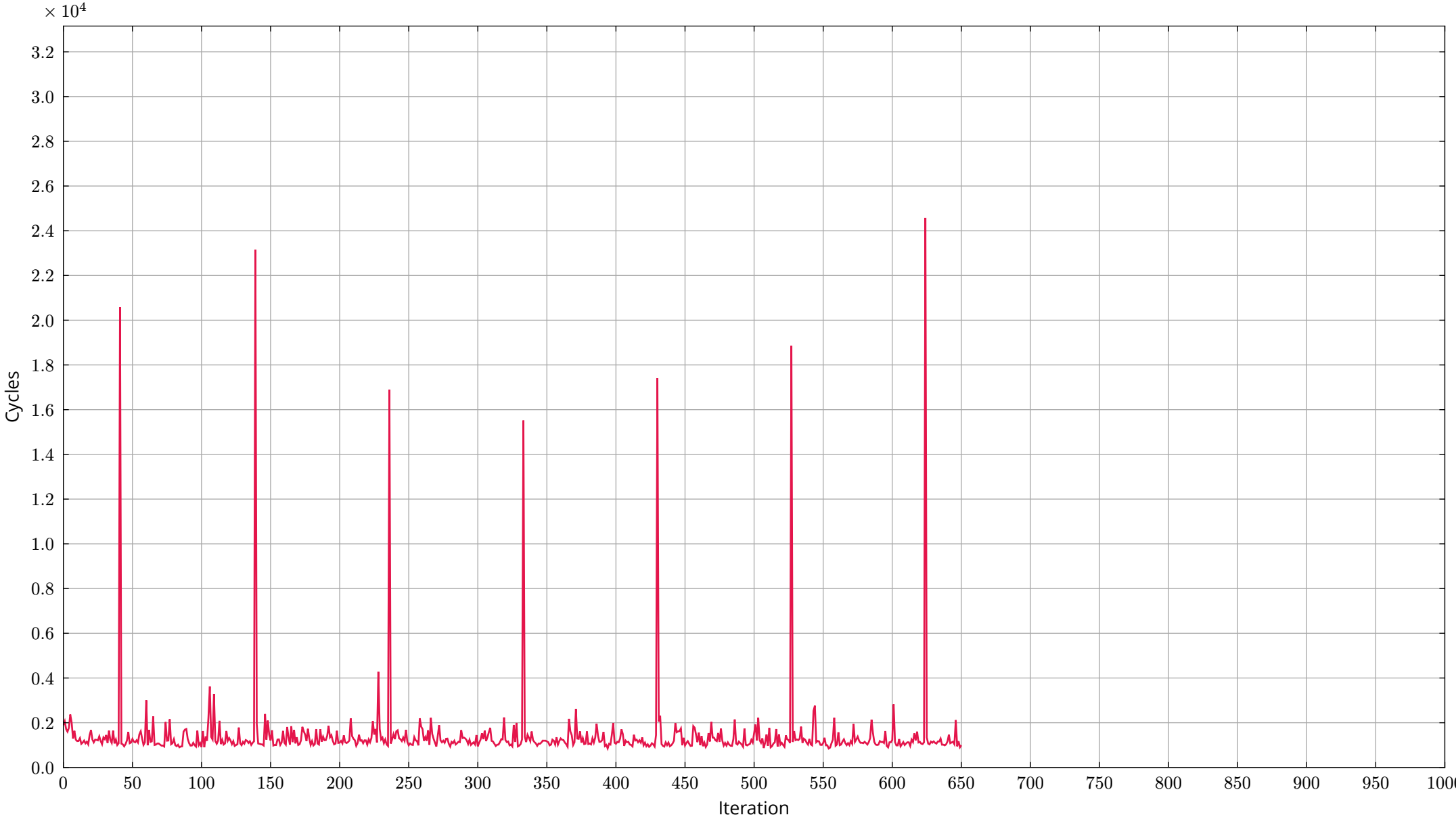

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration

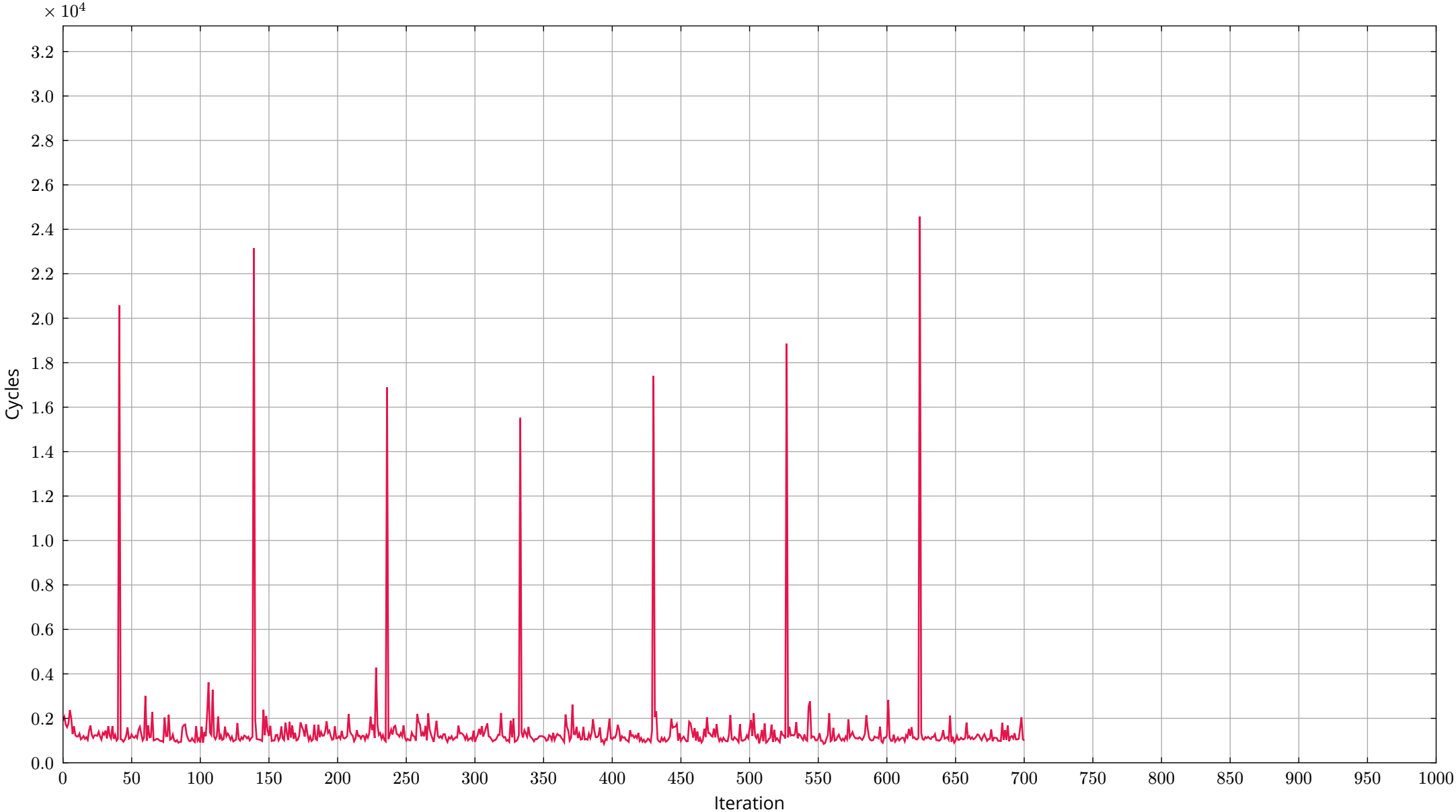

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration

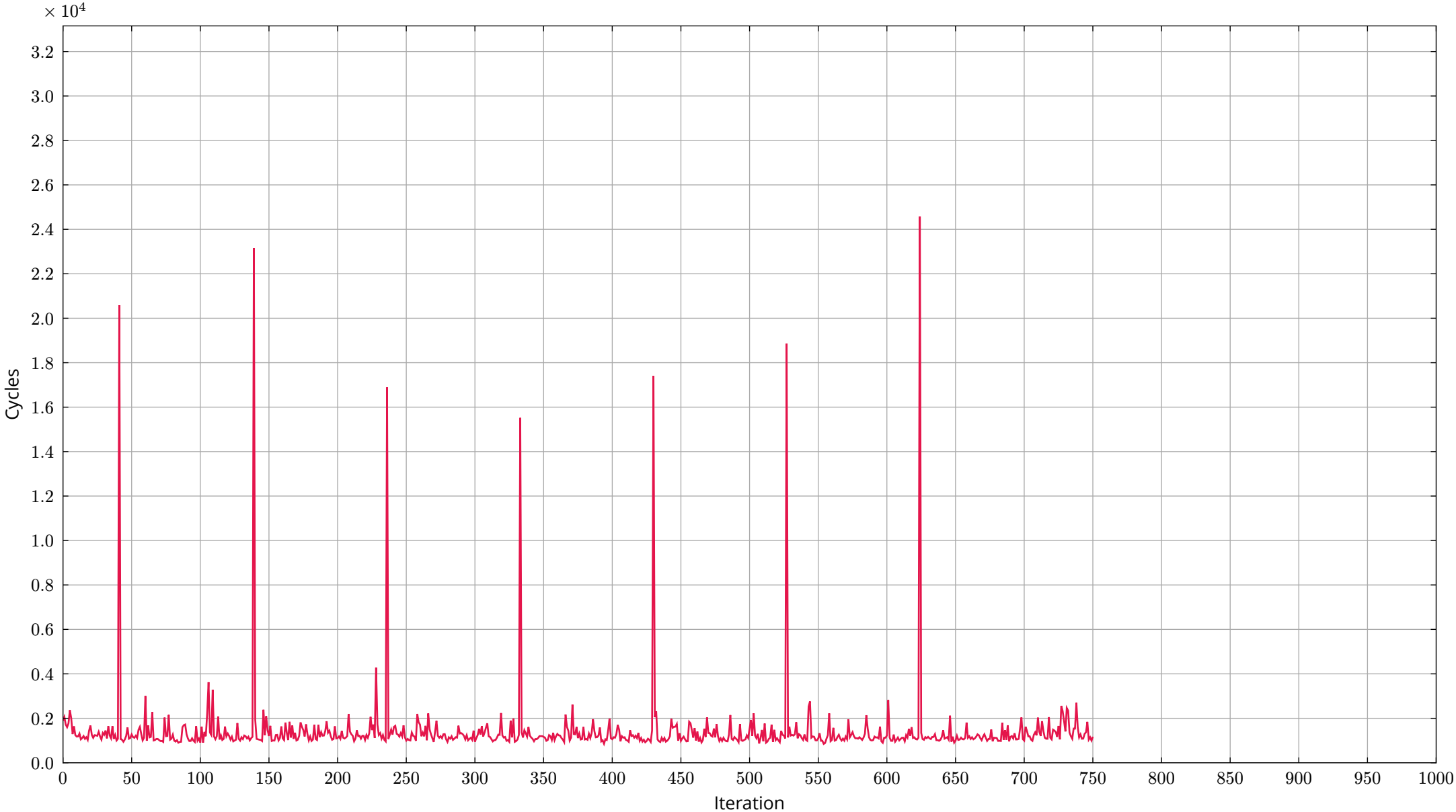

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration

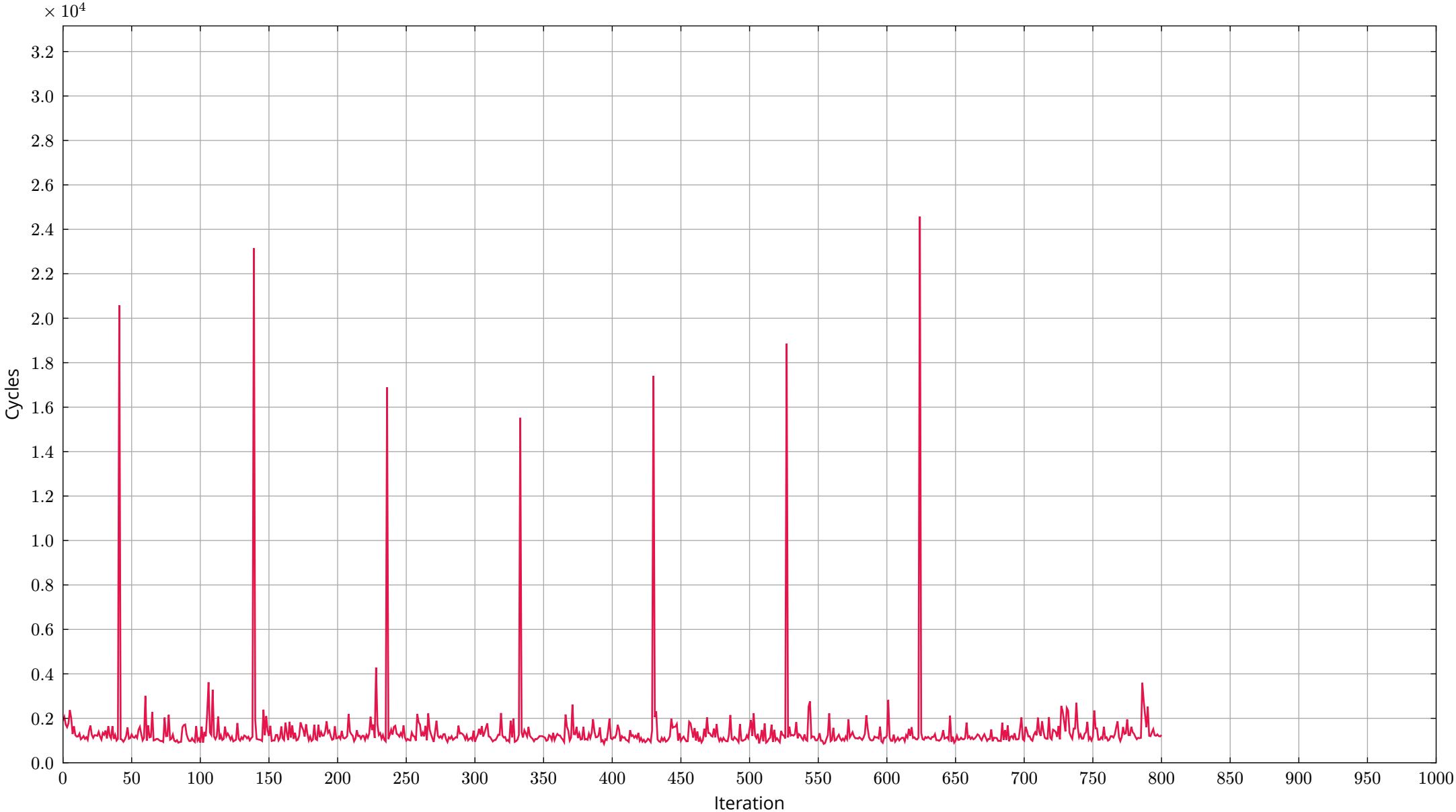

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration

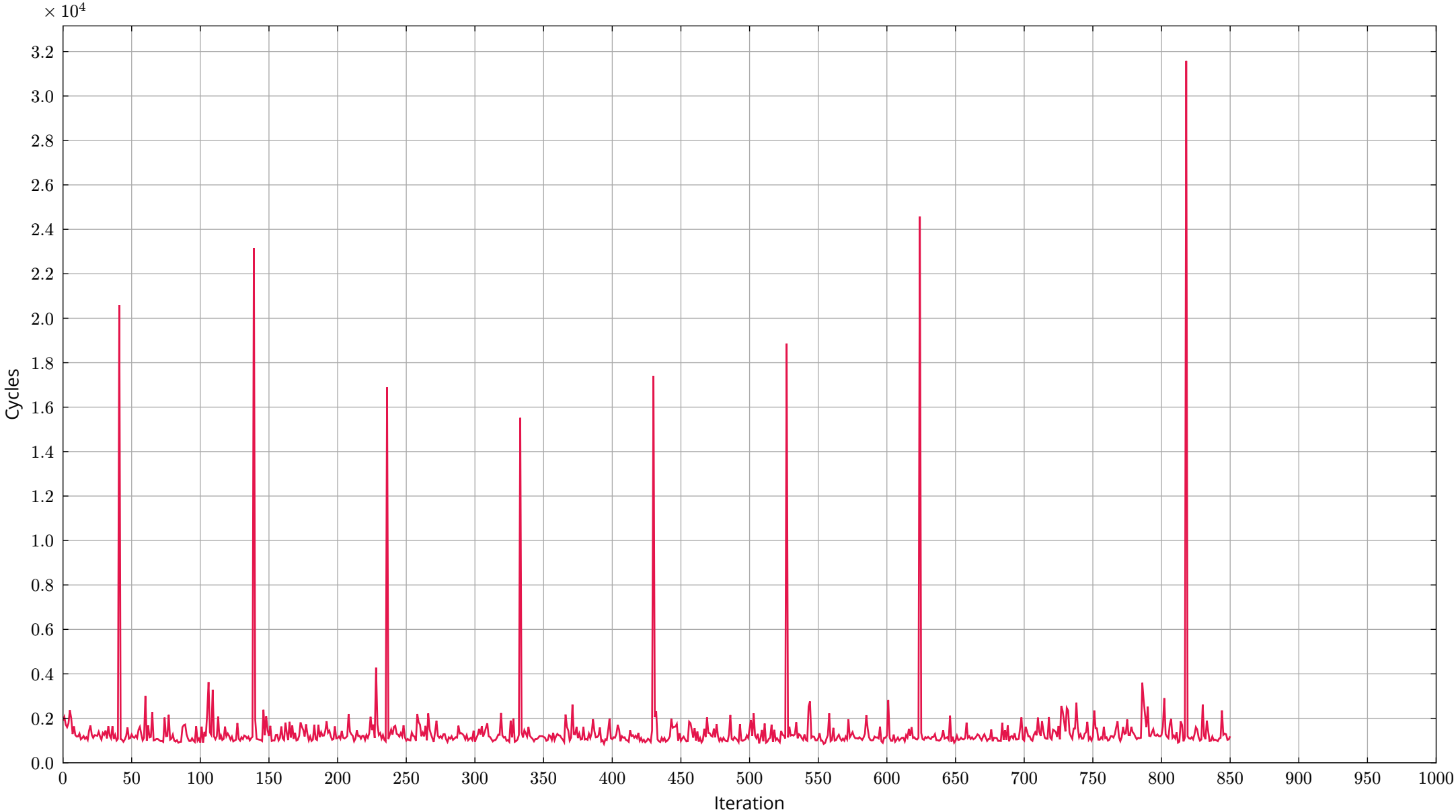

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration

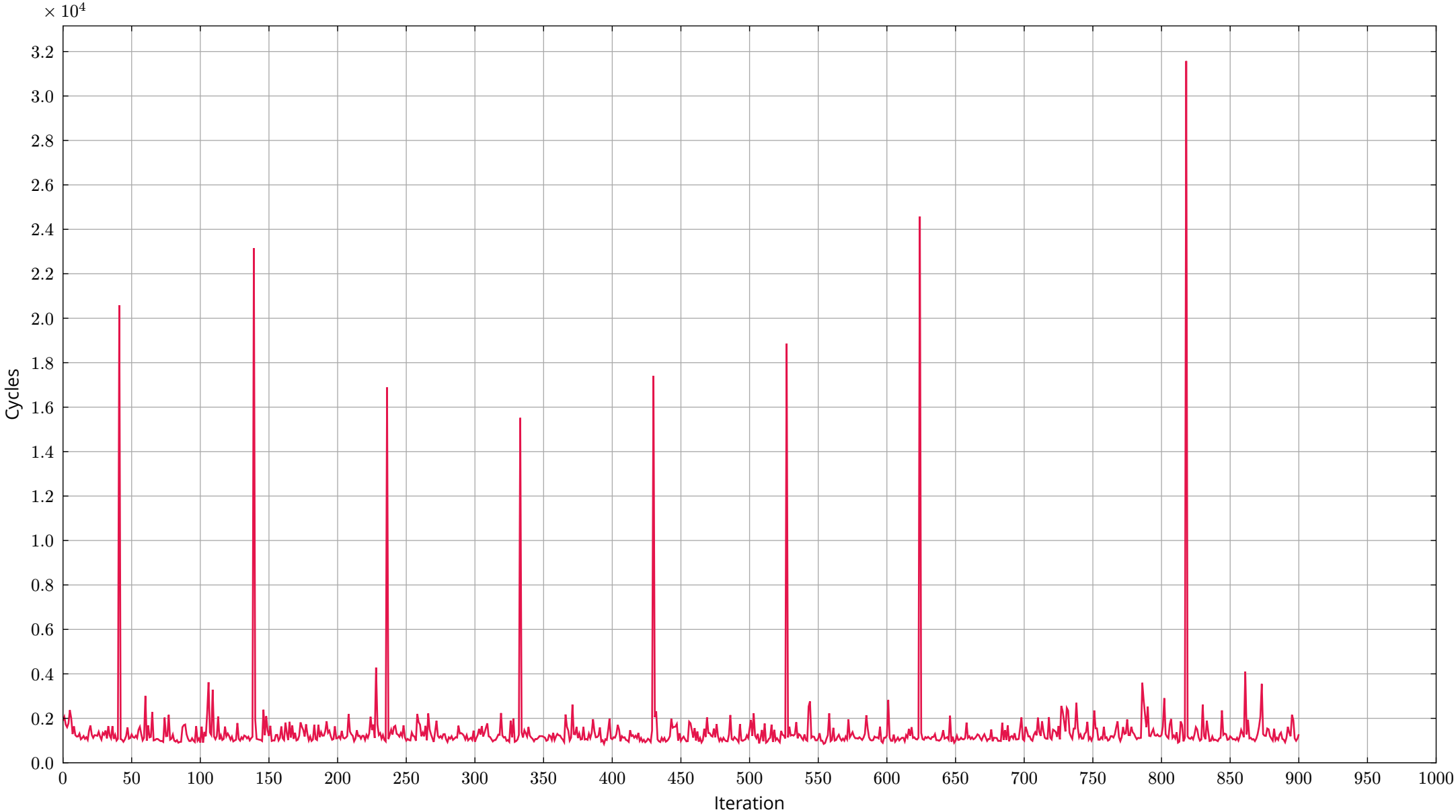

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration

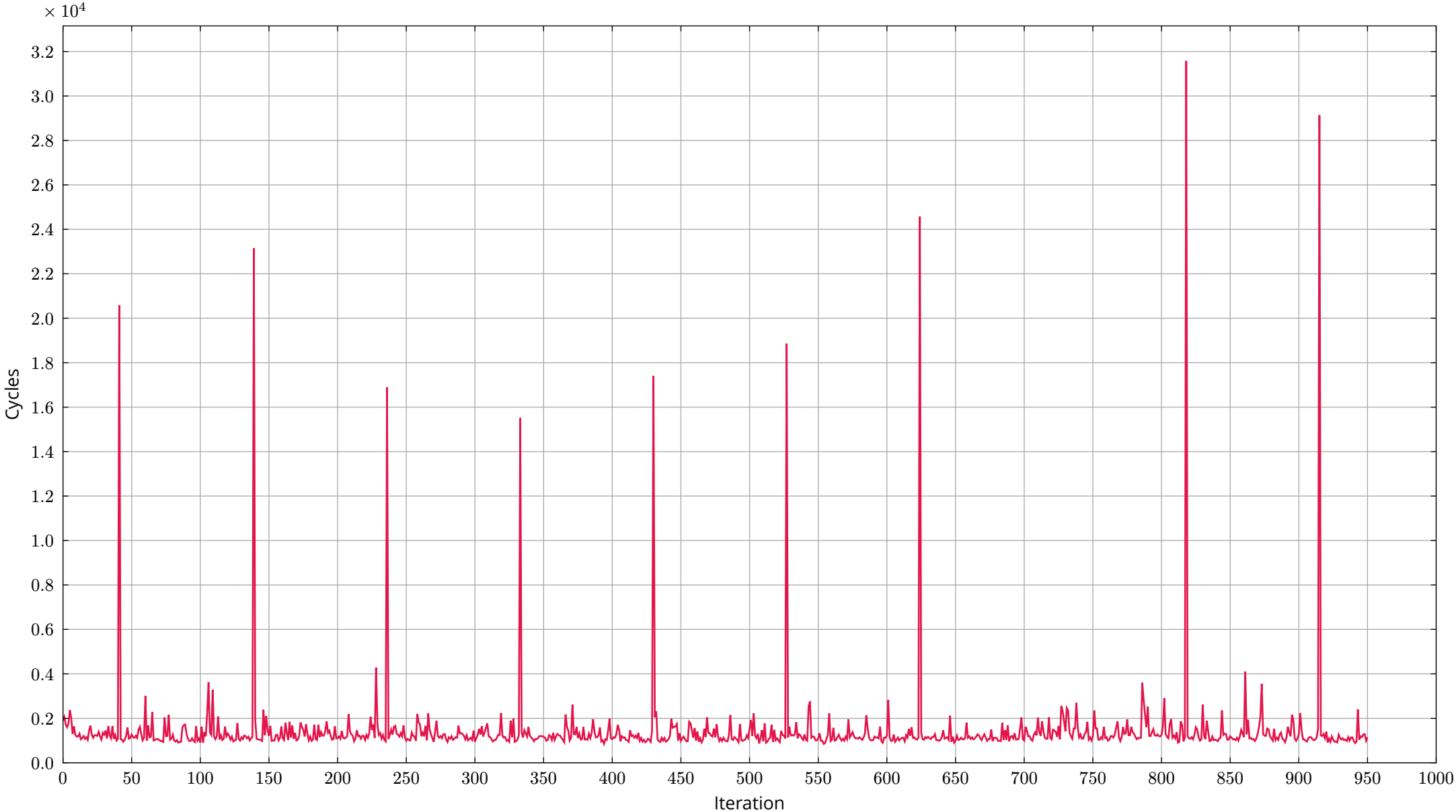

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration

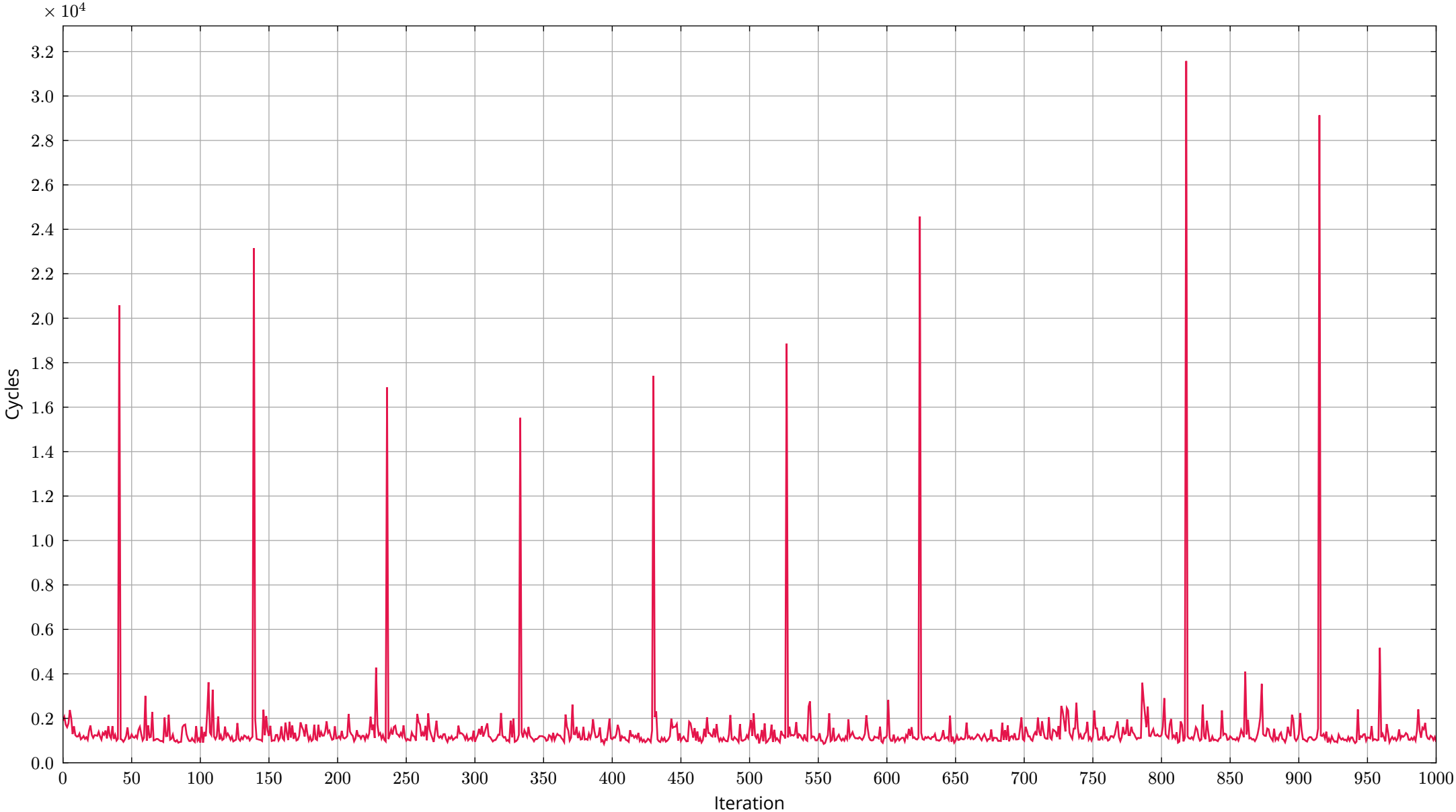

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration

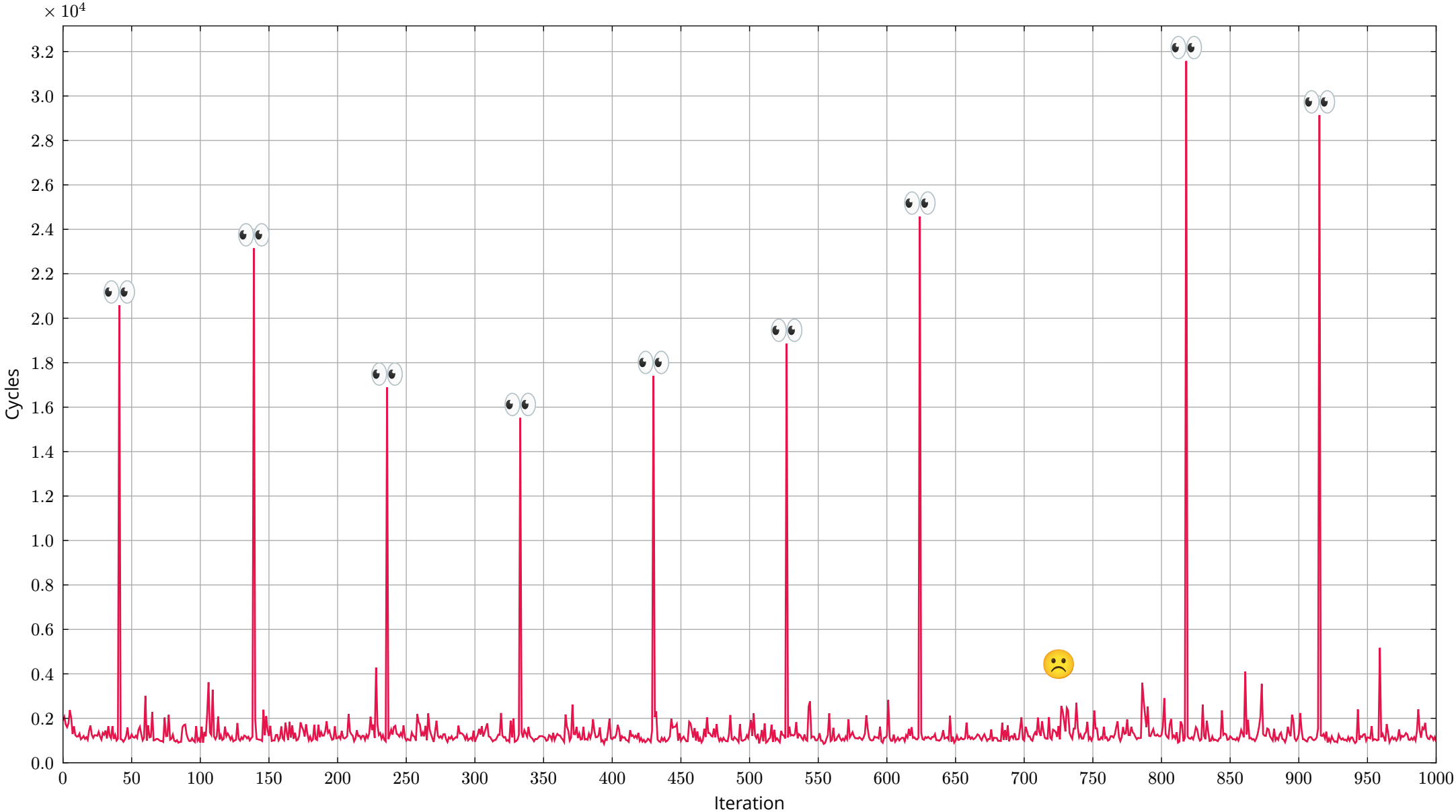

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration

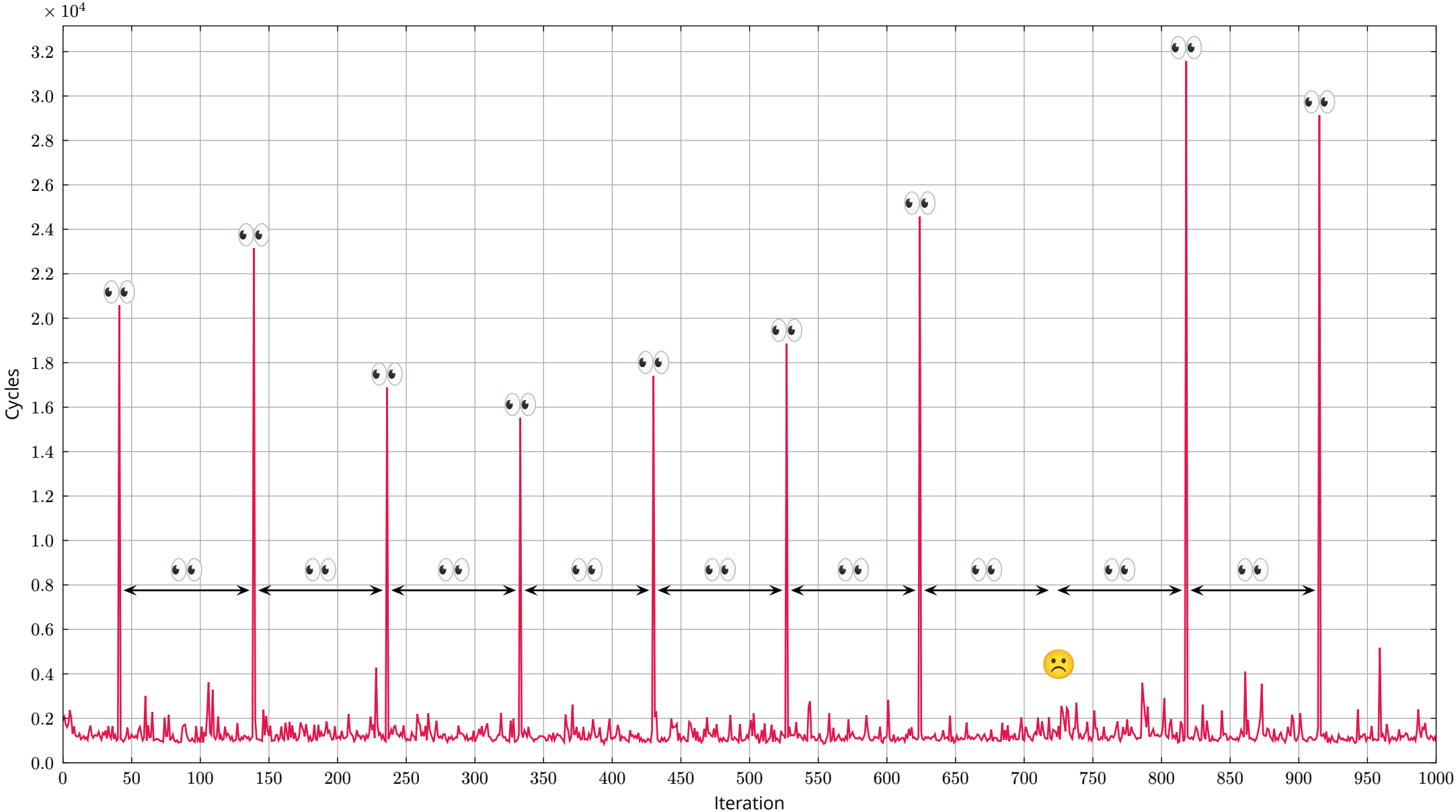

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration

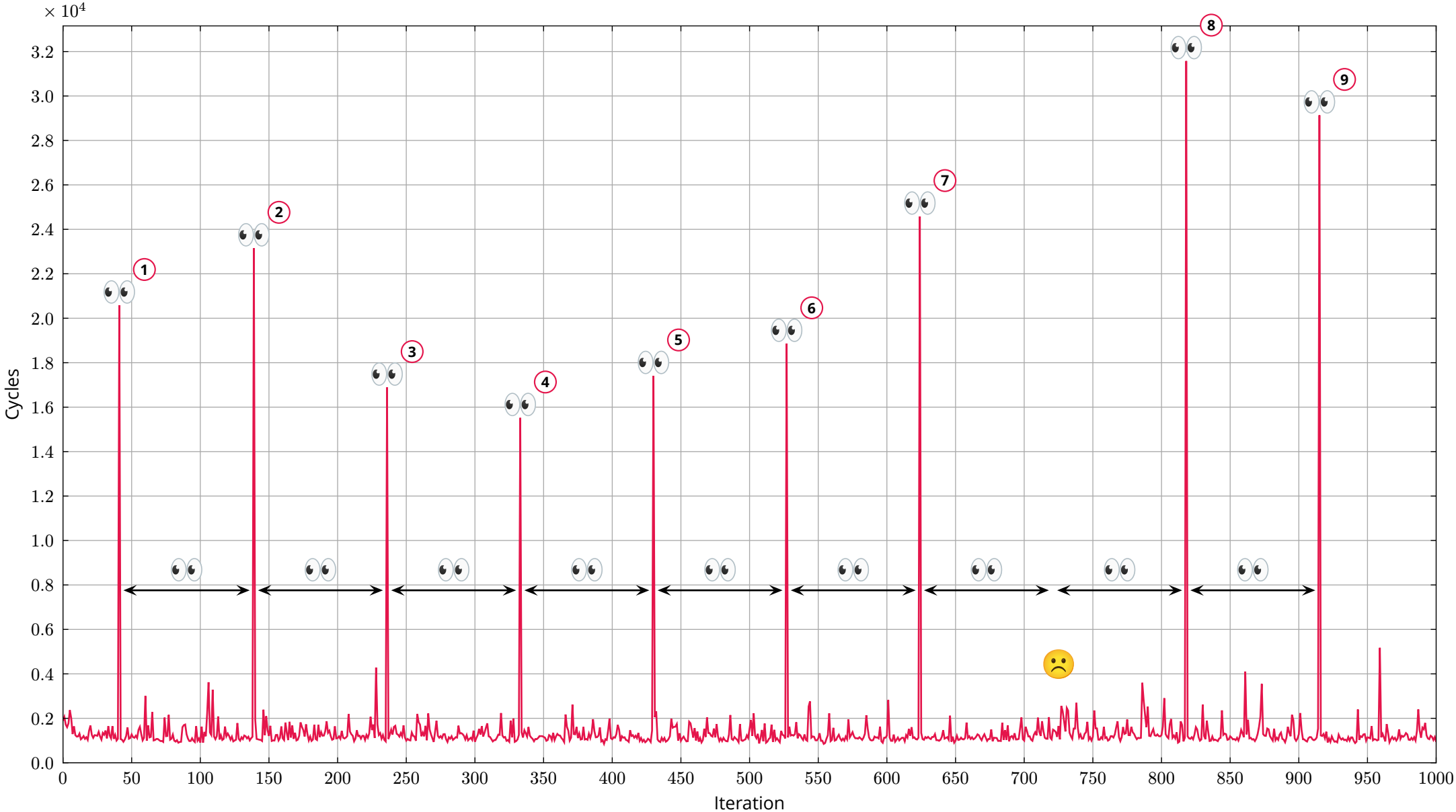

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration

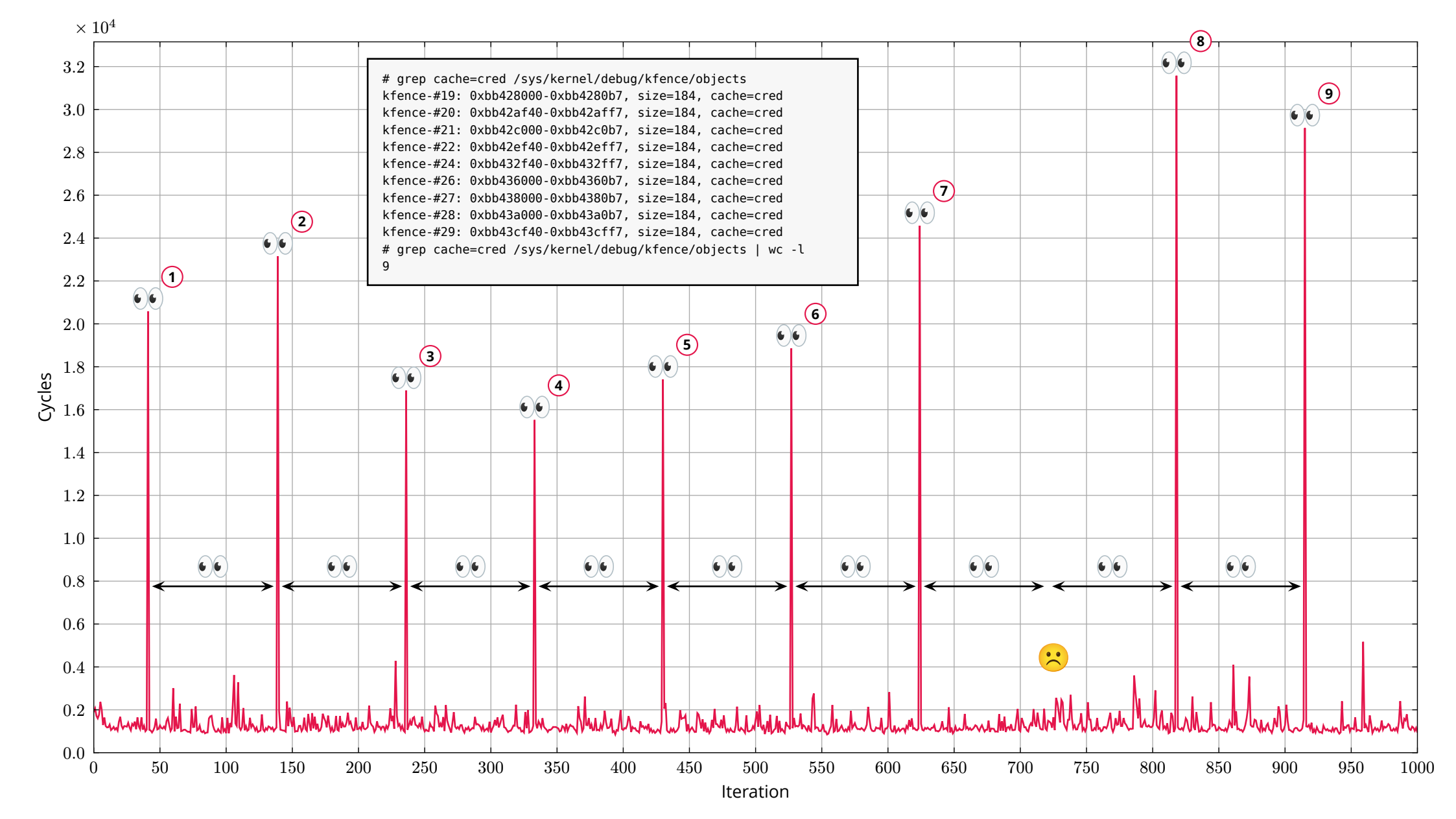

× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration


× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration


× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration


× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration


× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration
1
2
3
4
5
6
7
8
9


× 10^4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1000
Iteration
grep cache=cred /sys/kernel/debug/kfence/objects
kfence-#19: 0xbb428000-0xbb4280b7, size=184, cache=cred
kfence-#20: 0xbb42af40-0xbb42aff7, size=184, cache=cred
kfence-#21: 0xbb42c000-0xbb42c0b7, size=184, cache=cred
kfence-#22: 0xbb42ef40-0xbb42eff7, size=184, cache=cred
kfence-#24: 0xbb432f40-0xbb432ff7, size=184, cache=cred
kfence-#26: 0xbb436000-0xbb4360b7, size=184, cache=cred
kfence-#27: 0xbb438000-0xbb4380b7, size=184, cache=cred
kfence-#28: 0xbb43a000-0xbb43a0b7, size=184, cache=cred
kfence-#29: 0xbb43cf40-0xbb43cff7, size=184, cache=cred
grep cache=cred /sys/kernel/debug/kfence/objects | wc -l
9
1
2
3
4
5
6
7
8
9

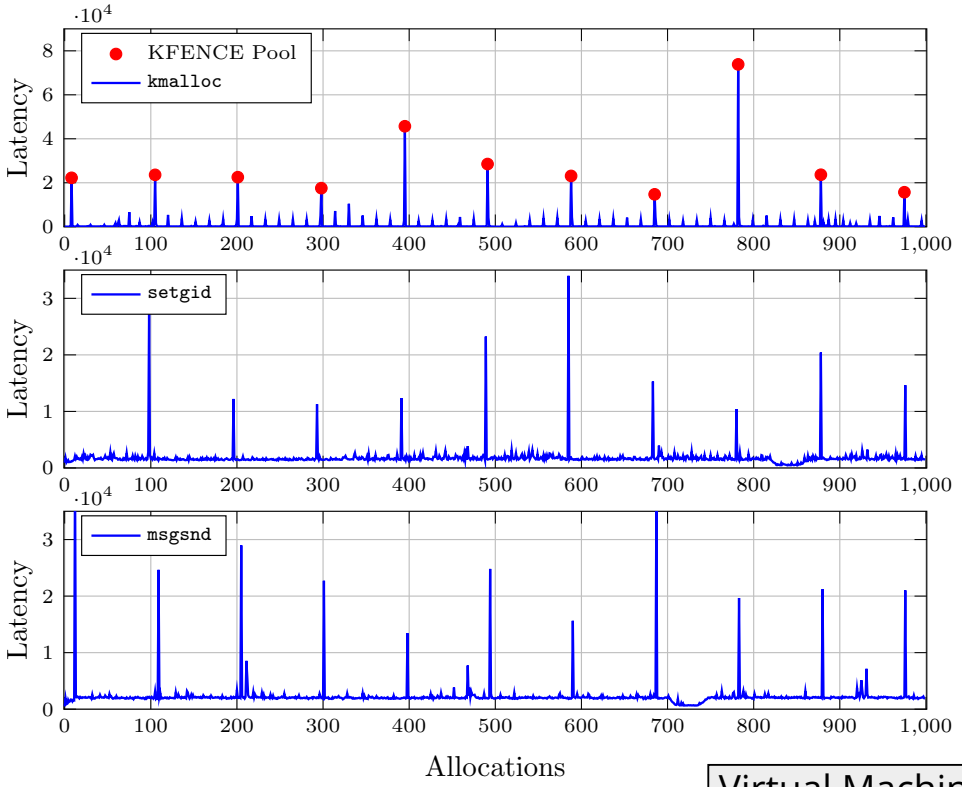
Timing Side Channel

Timing Side Channel

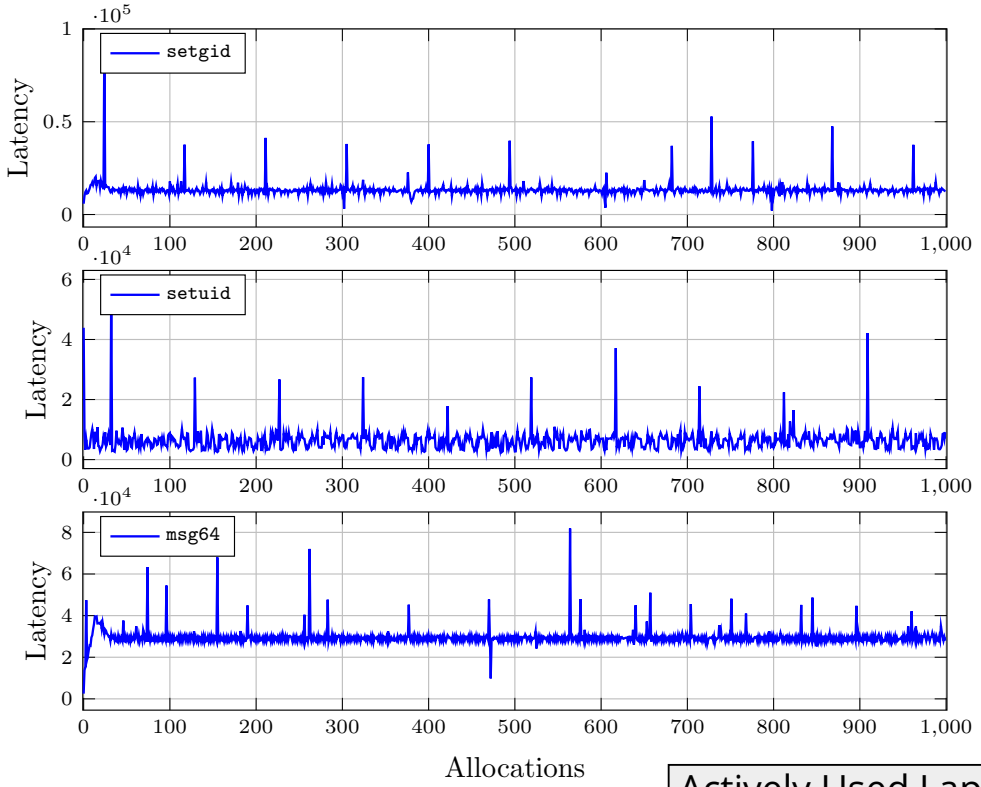
> Attacker may **pick KFENCE** allocations, **predict and spray** future windows

Timing Side Channel

Attacker may **pick KFENCE** allocations, **predict and spray** future windows



Virtual Machine



Actively Used Laptop

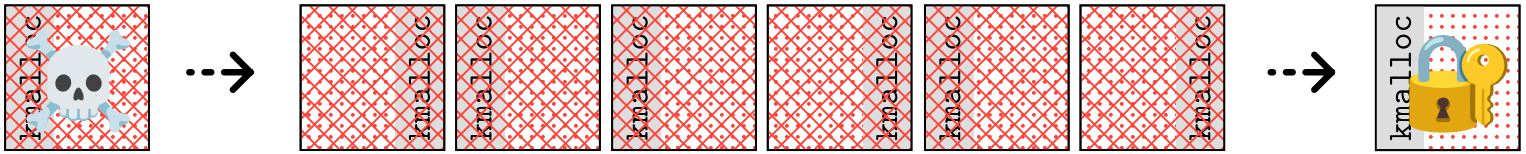
Freelist Estimation

Freelist Estimation

> Use-after-free reclaiming on a FIFO freelist requires guessing its length

Freelist Estimation

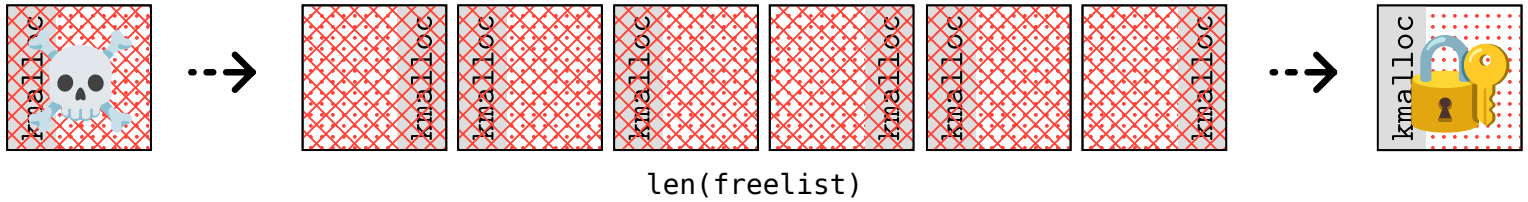
Use-after-free reclaiming on a FIFO freelist requires guessing its length




```
free(UAF) > (alloc + free) * N > alloc()
```

Freelist Estimation

Use-after-free reclaiming on a FIFO freelist requires guessing its length




```
free(UAF) > (alloc + free) * N > alloc()
```

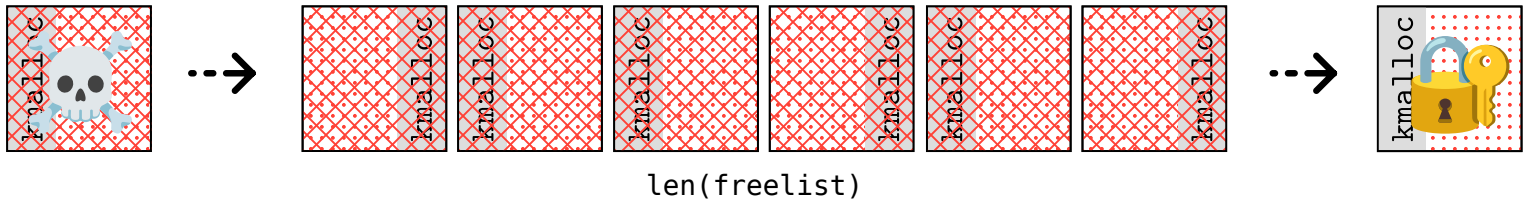
Option 1: Pressure Mode

If KFENCE stops serving same-callstack allocations, pressure mode has started

↳ This means 75% occupation → freelist 25% → 63 alloc + free

Freelist Estimation

> Use-after-free reclaiming on a FIFO freelist requires guessing its length




```
free(UAF) > (alloc + free) * N > alloc()
```

Option 1: Pressure Mode

If KFENCE stops serving same-callstack allocations, pressure mode has started

↳ This means 75% occupation → freelist 25% → 63 alloc + free

Option 2: Assume 25%

KFENCE tends to range around the pressure-mode cutoff → ~63 alloc + free

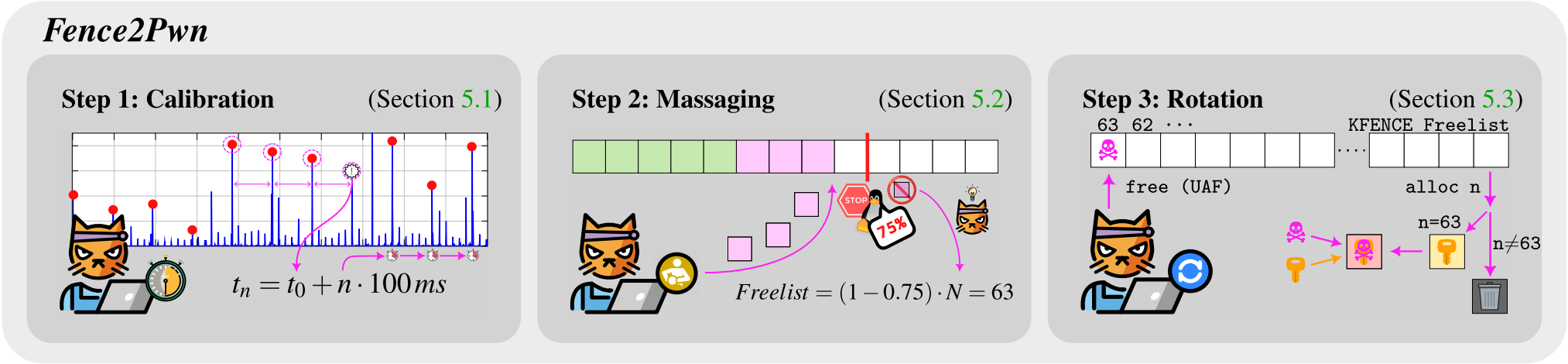
Fence2Pwn

Fence2Pwn

> A cache-agnostic object-overlapping technique that bypasses Slab

Fence2Pwn

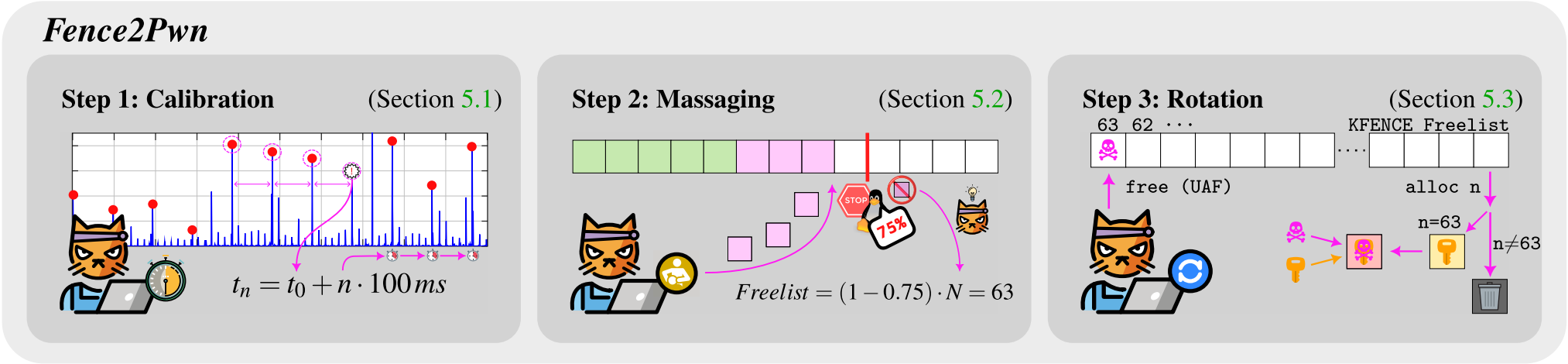
A cache-agnostic object-overlapping technique that bypasses Slab



Target KFENCE **Estimate len(freelist)** **Discard N + Reclaim**

Fence2Pwn

A cache-agnostic object-overlapping technique that bypasses Slab



Target KFENCE **Estimate len(freelist)** **Discard N + Reclaim**

Failed attempts due to non-overlapping KFENCE objects do not crash

↳ An **attacker can retry indefinitely** until good 1/2 random placement

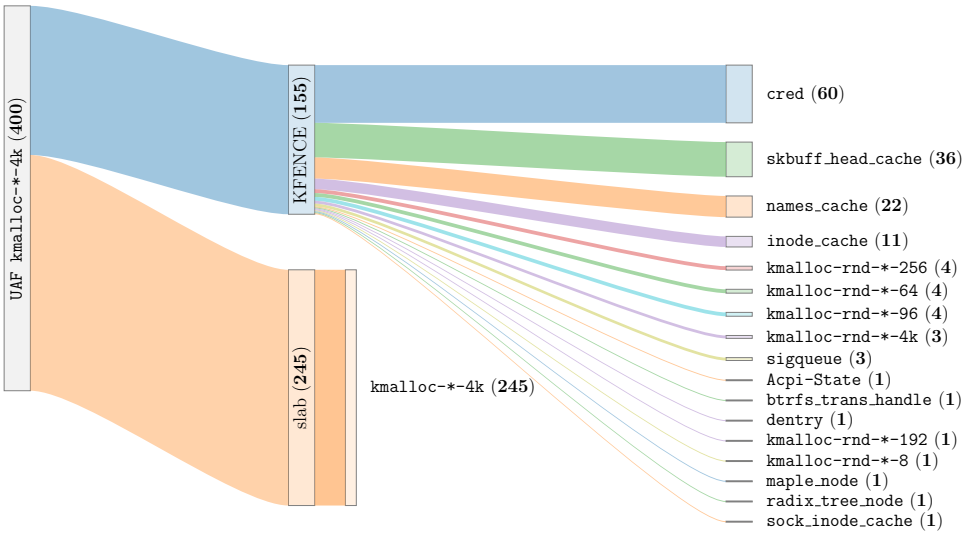
Kernel Object Reclaim

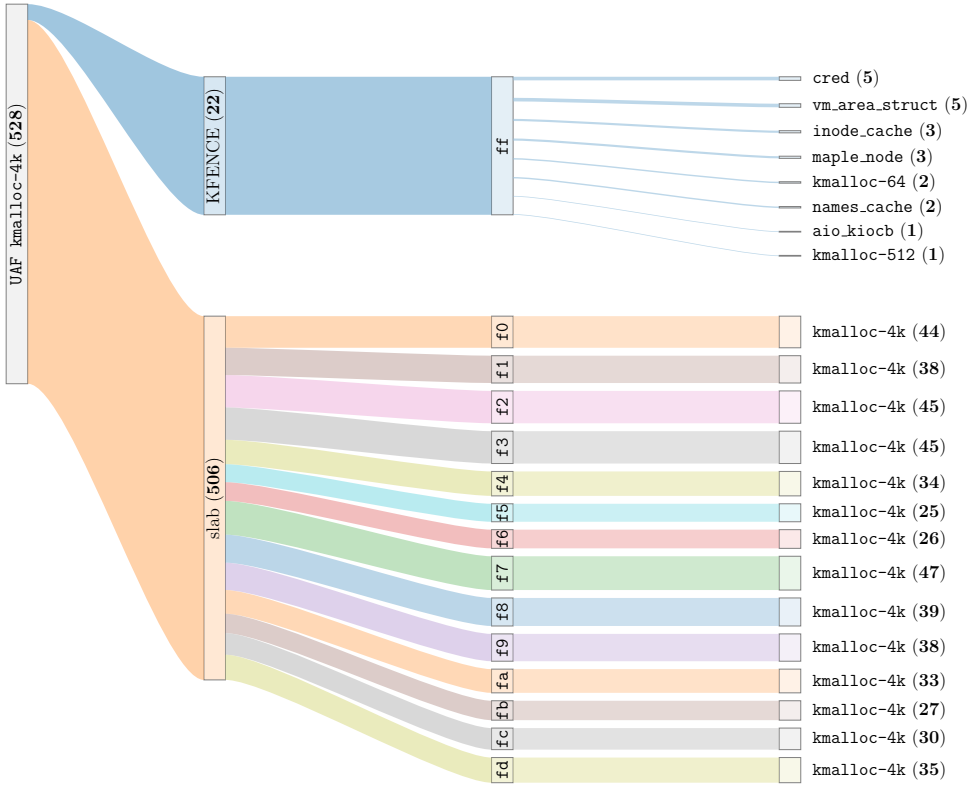
Kernel Object Reclaim

Object type diagram of **reclaimed memory** from previous synthetic **UAF**

Kernel Object Reclaim

Object type diagram of **reclaimed memory** from previous synthetic **UAF**





Bypass of Slab Hardening

↳ Heap Segregation (+ `SLAB_VIRTUAL`)

↳ Memory Tagging

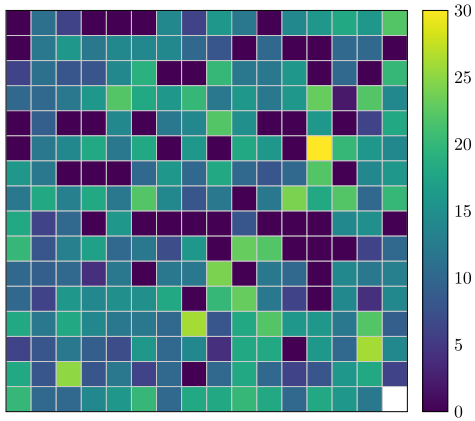
Naive Spray-based Exploitation

Naive Spray-based Exploitation

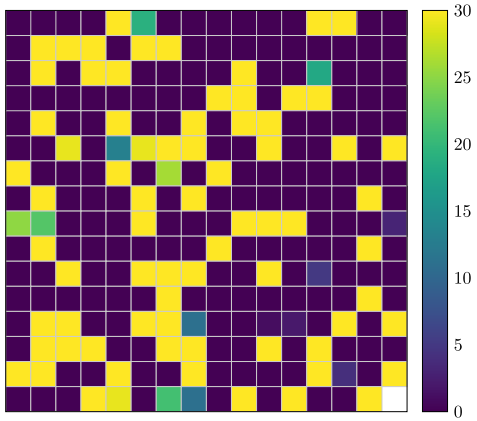
> High-uptime systems **concentrate reclaim activity** on fewer slots

Naive Spray-based Exploitation

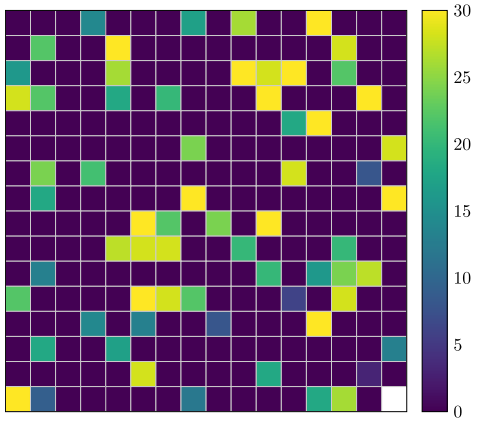
High-uptime systems **concentrate reclaim activity** on fewer slots



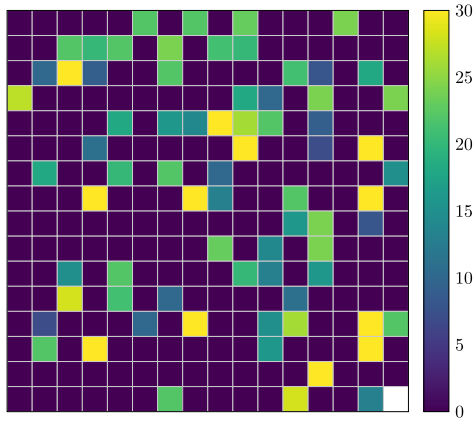
(a) Server A (13h uptime)



(b) Laptop (3d uptime)



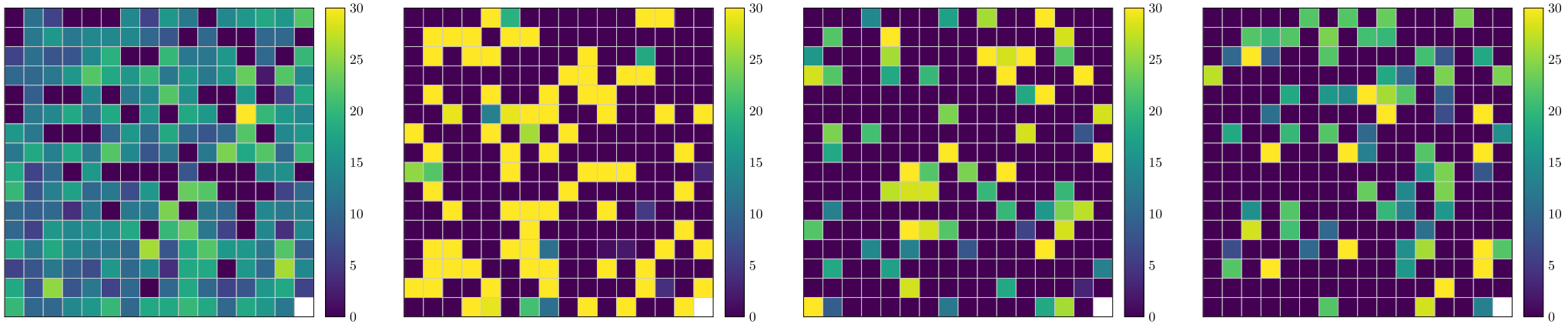
(c) Server B (7d uptime)



(d) Laptop (9d uptime)

Naive Spray-based Exploitation

High-uptime systems **concentrate reclaim activity** on fewer slots



(a) Server A (13h uptime) (b) Laptop (3d uptime) (c) Server B (7d uptime) (d) Laptop (9d uptime)

Long-lived objects hoard KFENCE memory slots as they do not deallocate

↳ Allocation and deallocations concentrates on few slots

↳ More likely to **reclaim** target **UAF** object slots during **sprays**

Conclusion



Upstream Patches

Upstream Patches

- Upstream implemented three patches

Upstream Patches

Upstream implemented three patches

1. Disabling KFENCE on MTE

↳ MTE disables KFENCE at boot time

Upstream Patches

Upstream implemented three patches

1. Disabling KFENCE on MTE

↳ MTE disables KFENCE at boot time

2. Dedicated crash-on-fault toggles

↳ Better KFENCE fault controls

Upstream Patches

> Upstream implemented three patches

1. Disabling KFENCE on MTE

↳ MTE disables KFENCE at boot time

2. Dedicated crash-on-fault toggles

↳ Better KFENCE fault controls

3. Freelist Fisher-Yates Shuffle

↳ Further freelist randomization

Upstream Patches

> Upstream implemented three patches

1. Disabling KFENCE on MTE

↳ MTE disables KFENCE at boot time

2. Dedicated crash-on-fault toggles

↳ Better KFENCE fault controls

3. Freelist Fisher-Yates Shuffle

↳ Further freelist randomization

Heap segregation remains unsolved

Upstream Patches

Upstream implemented three patches

1. Disabling KFENCE on MTE

↳ MTE disables KFENCE at boot time

2. Dedicated crash-on-fault toggles

↳ Better KFENCE fault controls

3. Freelist Fisher-Yates Shuffle

↳ Further freelist randomization

Heap segregation remains unsolved



Upstream Patches

Upstream implemented three patches

1. Disabling KFENCE on MTE

↳ MTE disables KFENCE at boot time

2. Dedicated crash-on-fault toggles

↳ Better KFENCE fault controls

3. Freelist Fisher-Yates Shuffle

↳ Further freelist randomization

Heap segregation remains unsolved



Upstream Patches

Upstream implemented three patches

1. Disabling KFENCE on MTE

↳ MTE disables KFENCE at boot time

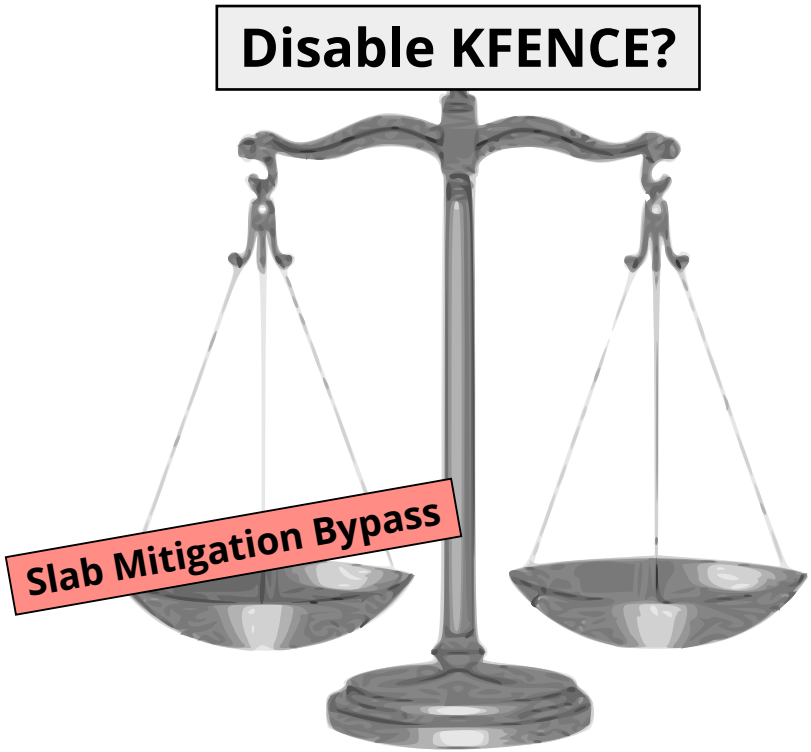
2. Dedicated crash-on-fault toggles

↳ Better KFENCE fault controls

3. Freelist Fisher-Yates Shuffle

↳ Further freelist randomization

Heap segregation remains unsolved



Upstream Patches

Upstream implemented three patches

1. Disabling KFENCE on MTE

↳ MTE disables KFENCE at boot time

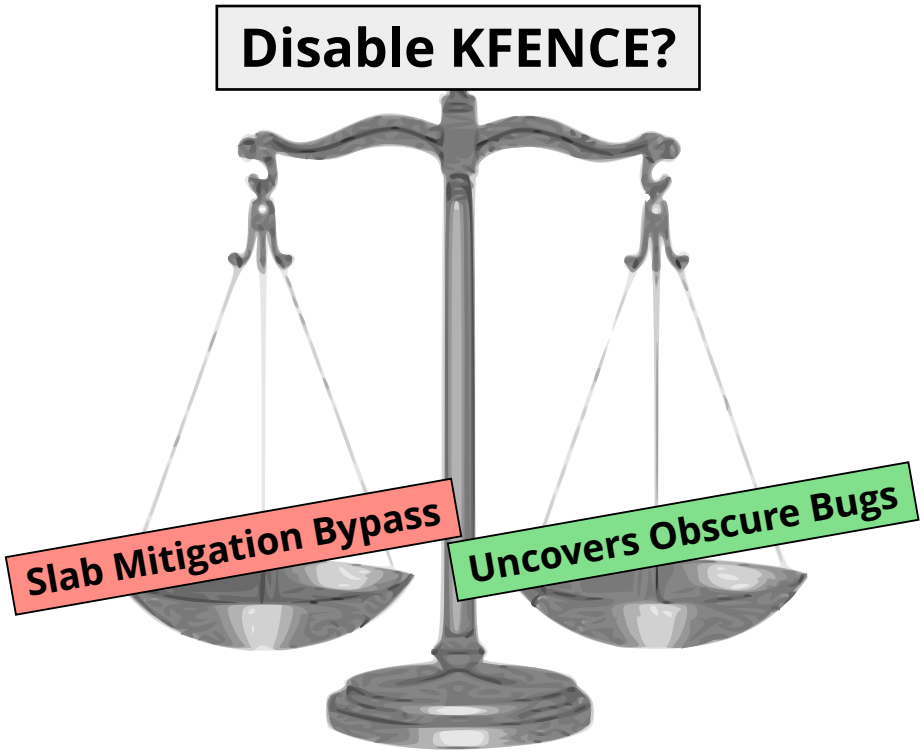
2. Dedicated crash-on-fault toggles

↳ Better KFENCE fault controls

3. Freelist Fisher-Yates Shuffle

↳ Further freelist randomization

Heap segregation remains unsolved



Open Problems

Open Problems

> No clean solution to solve the heap segregation problem

Open Problems

No clean solution to solve the heap segregation problem

$$(255 + 1) \times 2 \times 4096 = 2\text{MiB}$$

Open Problems

> No clean solution to solve the heap segregation problem

$$(255 + 1) \times 2 \times 4096 = 2\text{MiB}$$

Scaling to one pool per Slab cache is unreasonable: 500x overhead

↳ Static smaller pools may waste memory and not scale

↳ Dynamic-sized pools may lead to cross-cache attacks in KFENCE

Open Problems

> No clean solution to solve the heap segregation problem

$$(255 + 1) \times 2 \times 4096 = 2\text{MiB}$$

Scaling to one pool per Slab cache is unreasonable: 500x overhead

↳ Static smaller pools may waste memory and not scale

↳ Dynamic-sized pools may lead to cross-cache attacks in KFENCE

Mantainability

- With two allocators an attacker picks the **weakest link**
- Both must be **equally secure**
- What if there is new Slab hardening techniques in the future?

TU Graz

SCIENCE
PASSION
TECHNOLOGY

Security Implications of Linux Kernel Electric Fence

Studying and Attacking KFENCE

Ernesto Martínez García

Linux Security Summit Europe 2026

› isec.tugraz.at

Appendix



Download Article

- Download Article <
- Download Slides
- Download Infographic
- Website
- Email

Ernesto Martínez García

PhD Student @ Graz University of Technology

ernesto.martinezgarcia@tugraz.at

ernestomartinezgarcia.com

linkedin.com/in/ecomaikgolf

Looking for cool Linux security research projects/ideas :)

Appendix



- Download Article
- Download Slides <
- Download Infographic
- Website
- Email

Ernesto Martínez García

PhD Student @ Graz University of Technology

ernesto.martinezgarcia@tugraz.at

ernestomartinezgarcia.com

linkedin.com/in/ecomaikgolf

Looking for cool Linux security research projects/ideas :)

Appendix



Download Infogr.

- Download Article
- Download Slides
- Download Infographic <
- Website
- Email

Ernesto Martínez García

PhD Student @ Graz University of Technology

ernesto.martinezgarcia@tugraz.at

ernestomartinezgarcia.com

linkedin.com/in/ecomaikgolf

Looking for cool Linux security research projects/ideas :)

Appendix



- Download Article
- Download Slides
- Download Infographic
- Website <
- Email

Ernesto Martínez García

PhD Student @ Graz University of Technology

ernesto.martinezgarcia@tugraz.at

ernestomartinezgarcia.com

linkedin.com/in/ecomaikgolf

Looking for cool Linux security research projects/ideas :)

Appendix



Email

- Download Article
- Download Slides
- Download Infographic
- Website
- Email <

Ernesto Martínez García

PhD Student @ Graz University of Technology

ernesto.martinezgarcia@tugraz.at

ernestomartinezgarcia.com

linkedin.com/in/ecomaikgolf

Looking for cool Linux security research projects/ideas :)