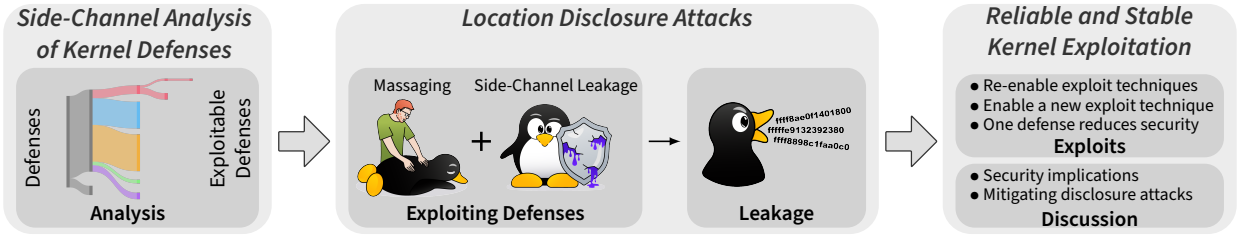
When Good Kernel Defenses Go Bad:

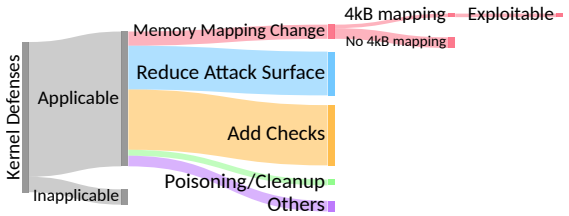
Reliable and Stable Kernel Exploits via Defense-Amplified TLB Side-Channel Leaks

Lukas Maar[1] Lukas Giner[1] Daniel Gruss[1] Stefan Mangard[1]

TU Graz
SCIENCE
PASSION
TECHNOLOGY



Side-Channel Analysis of Kernel Defenses



Enforcing strict kernel memory permissions or virtualizing the kernel heap or stack amplifies TLB leakage.

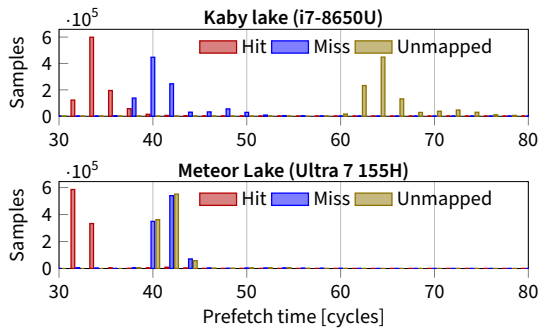
Massaging + Side-Channel Leakage


```
sys_msgrcv(id, mtext, mtype):
  msq = ipc_ns.root_rt[id]
  if !msg:
    return
  msg = find_msg(msq, mtype)
  copy_to_user(mtext, msg.mtext)
```

```
mtext = char[]
mtype = 0x41

// access msg0, msq0, ipc_ns
sys_msgrcv(0, mtext, mtype) ①

// access msg1, msq1, ipc_ns
sys_msgrcv(1, mtext, mtype) ②

// access msg32, msq32, ipc_ns
sys_msgrcv(32, mtext, mtype) ③
```

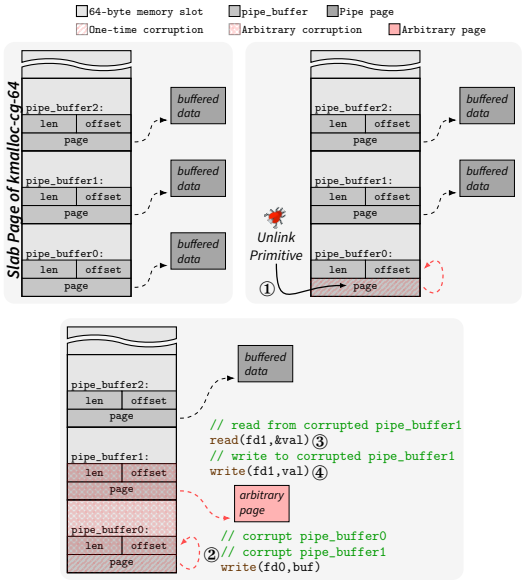
Strategic massaging of the kernel allocator allows to create TLB contention pattern to leak the page ❷ location.

Location Disclosure Attacks



Three TLB distinguishing primitives: *hit/miss*, *mapped/unmapped*, and *2 MB/4 kB primitive*.

Reliable and Stable Kernel Exploitation



Three side-channel-assisted exploit techniques: converting an unlink primitive to arbitrary r/w (shown here), and converting an arbitrary free to arbitrary r/w and constrained write to code execution (in the paper).

Key Takeaways

1. Side-channel leakage can increase the reliability of kernel exploits, approaching nearly 100 % without system crashes.
2. While kernel defenses enhance security in one dimension, some may unintentionally weaken it in another.
3. Some design decisions in the kernel memory allocator also may unintentionally weaken security.

Contact

Lukas Maar
lukas.maar@tugraz.at
https://lukasmaar.github.io
https://github.com/lukasmaar

[1]Graz University of Technology
USENIX Security 2025 | lukas.maar@tugraz.at

ISEC | Institute of Information Security
› isec.tugraz.at