PeriScope: An Effective Probing and Fuzzing Framework for the Hardware-OS Boundary

- Finding bugs in OS drivers using fuzzing.
- Limitations of previous fuzzing work
  - Require developer effort tailored to specific drivers and devices (either physical or virtual)
  - Only target at interruptions.
- Threat model
  - Peripheral devices are compromised through known bugs.
  - Devices can send arbitrary data to device drivers.
  - Devices only access the memory regions used for communication with device drivers.
- Contributions
  - Support Memory-Mapped I/O and Direct Memory Access.
  - Hook the kernel's page fault handler and make it driver-agnostic.
    - Monitor the drivers when they create MMIO or DMA memory mapping.
    - Mark the pages as unpresented to raise page fault.
    - Monitor the diver READ operations and fuzz them.

Fuzzing Kernel

- Fuzzer repeatedly hit known shallow bugs (the OS crashed and reboots).
  - “We circumvented this problem by disabling certain code paths that contain previously discovered shallow bugs. This does, however, somewhat reduce the effectiveness of our fuzzer as it cannot traverse the subpaths rooted at these blacklisted bugs.”
- Whole-system fuzzers fuzz the system without restarting between fuzzing iterations. Internal states previously set affect the runs after.

Loosely typed nature of PHP

- Implicit type conversion
  - $var = 1 . “test”;
- Type juggling
  - In compar operations
  - “1test” == 1 returns **true** in php loose comparison
  - There exists CVEs about authentication bypass.

- Server-side web security
  - Find something special in PHP or in server-side applications.
- OS
  - Driver, file system.
- Architecture.
  - Transient Execution