Contribution ID: **90** Type: **not specified**

Compiler Features for Kernel Security

Friday, 24 September 2021 10:05 (30 minutes)

GCC and Clang both have a variety of security features available, but they are not always at parity with each other. This discussion will review the security features important to the Linux kernel with regard to what's working, what's missing, and what needs adjustment.

Specifically, these areas will be discussed along with anything else that seems relevant:

- stack protector guard location (i.e. enabling per-task canaries)

```
-mstack-protector-guard=sysreg
-mstack-protector-guard-reg=sp\_el0
-mstack-protector-guard-offset=0
```

- call-used register zeroing (now in GCC 11)

```
-fzero-call-used-regs
```

- stack variable auto-initialization (already in Clang, soon to be in GCC 12)

```
-ftrivial-auto-var-init=\{zero,pattern\}
```

- array bounds checking

```
-Warray-bounds
-Wzero-length-bounds
-Wzero-length-array
-fsanitize=bounds
-fsanitize=bounds-strict
```

- integer overflow protection

```
-fsanitize=signed-integer-overflow
-fsanitize=unsigned-integer-overflow
```

- Link Time Optimization

```
-flto
-flto=thin
```

- backward edge Control Flow Integrity

```
-mbranch-protection=pac-ret[+leaf]
-fsanitize=shadow-call-stack
CET
```

- forward edge Control Flow Integrity

```
-fcf-protection=branch
-mbranch-protection=bti
-fsanitize=cfi
```

- Spectre v1 mitigation

```
-mspeculative-load-hardening
```

- structure layout randomization

```
\_\_attribute\_\_((randomize\_layout))
```

- constant expression for "is an lvalue?"
- constant expression for lvalue type extraction

I agree to abide by the anti-harassment policy

I agree

Primary authors: COOK, Kees (Google); ZHAO, Qing

Presenters: COOK, Kees (Google); ZHAO, Qing

Session Classification: Toolchains and Kernel MC

Track Classification: Toolchains and Kernel MC