The Undefined Behavior Sanitizer - UBSAN

UBSAN is a runtime undefined behaviour checker.

UBSAN uses compile-time instrumentation to catch undefined behavior (UB). Compiler inserts code that perform certain kinds of checks before operations that may cause UB. If check fails (i.e. UB detected) __ubsan_handle_* function called to print error message.

GCC has that feature since 4.9.x [1] (see `-fsanitize=undefined` option and its suboptions). GCC 5.x has more checkers implemented [2].

Report example

```
================================================================================
UBSAN: Undefined behaviour in ../include/linux/bitops.h:110:33
shift exponent 32 is to large for 32-bit type 'unsigned int'
CPU: 0 PID: 0 Comm: swapper Not tainted 4.4.0-rc1+ #26
 0000000000000000 ffffffff82403cc8 ffffffff815e6cd6 0000000000000001
 ffffffff82403cf8 ffffffff82403ce0 ffffffff8163a5ed 0000000000000020
 ffffffff82403d78 ffffffff8163ac2b ffffffff815f0001 0000000000000002
Call Trace:
 [<ffffffff815e6cd6>] dump_stack+0x45/0x5f
 [<ffffffff8163a5ed>] ubsan_epilogue+0xd/0x40
 [<ffffffff8163ac2b>] __ubsan_handle_shift_out_of_bounds+0xeb/0x130
 [<ffffffff815f0001>] ? radix_tree_gang_lookup_slot+0x51/0x150
 [<ffffffff8173c586>] _mix_pool_bytes+0x1e6/0x480
 [<ffffffff83105653>] ? dmi_walk_early+0x48/0x5c
 [<ffffffff8173c881>] add_device_randomness+0x61/0x130
 [<ffffffff83105b35>] ? dmi_save_one_device+0xaa/0xaa
 [<ffffffff83105653>] dmi_walk_early+0x48/0x5c
 [<ffffffff831066ae>] dmi_scan_machine+0x278/0x4b4
 [<ffffffff8111d58a>] ? vprintk_default+0x1a/0x20
 [<ffffffff830ad120>] ? early_idt_handler_array+0x120/0x120
 [<ffffffff830b2240>] setup_arch+0x405/0xc2c
 [<ffffffff830ad120>] ? early_idt_handler_array+0x120/0x120
 [<ffffffff830ae053>] start_kernel+0x83/0x49a
 [<ffffffff830ad120>] ? early_idt_handler_array+0x120/0x120
 [<ffffffff830ad386>] x86_64_start_reservations+0x2a/0x2c
 [<ffffffff830ad4f3>] x86_64_start_kernel+0x16b/0x17a
================================================================================
```

Usage

To enable UBSAN configure kernel with:

```
CONFIG_UBSAN=y
```

and to check the entire kernel:

```
CONFIG_UBSAN_SANITIZE_ALL=y
```

To enable instrumentation for specific files or directories, add a line similar to the following to the respective kernel Makefile:

- For a single file (e.g. main.o):

  ```
  UBSAN_SANITIZE_main.o := y
  ```

- For all files in one directory:

  ```
  UBSAN_SANITIZE := y
  ```

To exclude files from being instrumented even if `CONFIG_UBSAN_SANITIZE_ALL=y`, use:

```
UBSAN_SANITIZE_main.o := n
```

and:

```
UBSAN_SANITIZE := n
```

Detection of unaligned accesses controlled through the separate option - CONFIG_UBSAN_ALIGNMENT. It's off by default on architectures that support unaligned accesses (CONFIG_HAVE_EFFICIENT_UNALIGNED_ACCESS=y). One could still enable it in config, just note that it will produce a lot of UBSAN reports.

References