

Improving BIND 9 Code Quality

Why is concurrent programming so hard and what can we do about it?

Ondřej Surý, ISC
FOSDEM 2020

Prerequisites

- Continuous Integration
- Time to fix all the bugs

Some facts about BIND 9

- BIND 9:
 - Is 20+ years old
 - Is multithreaded
 - Uses locking (pthread_mutex)
 - Has own rwlock (that allows downgrade and upgrade)
- Since BIND 9.14+
 - Uses C11 stdatomic (or gcc __sync, or Windows Interlocked API)

Tools

- Static code analyzers
 - scan-build
 - cppcheck
 - Clang-tidy
- Runtime Analyzers
 - Address Sanitizer (ASAN)
 - Undefined Behaviour Sanitizer (UBSAN)
 - Thread Sanitizer (TSAN)
 - Memory Sanitizer (MSAN)
- Commercial (But Free) Tools
 - Coverity Scan
 - Sonar(?)
 - [LGTM.com](https://lgtm.com)
- Refactoring tools
 - coccinelle



Photo by [Barn Images](#) on [Unsplash](#)

scan-build

- Static Code Analyzer
- Replaces CC environment variables
- Comes from LLVM/Clang suite

scan-build usage

```
scan-build ./configure
```

```
scan-build --keep-cc make
```


scan-build examples

```
commit e9acad638eb21e0ef0bd8558a196ca24c3099292
```

```
Author: Ondřej Surý <ondrej@sury.org>
```

```
Date: Thu Oct 31 06:50:58 2019 -0500
```

```
libdns: add missing checks for return values in dnstap unit test
```

```
Related scan-build report:
```

```
dnstap_test.c:169:2: warning: Value stored to 'result' is never read
```

```
    result = dns_test_makeview("test", &view);
```

```
    ^~~~~~
```

```
dnstap_test.c:193:2: warning: Value stored to 'result' is never read
```

```
    result = dns_compress_init(&cctx, -1, dt_mctx);
```

```
    ^~~~~~
```

```
2 warnings generated.
```

Cppcheck

- Static code analyzer
- Requires compilation database
- Different levels of strictness
 - warning, performance, portability, information, missingInclude

Cppcheck usage

`./configure`

`bear make`

```
cppcheck --enable=warning,performance,portability,information,missingInclude \
         --include=config.h \
         --quiet \
         --std=c11 \
         --language=c \
         --project=compile_commands.json \
         --error-exitcode=2 \
         --xml \
         --output-file=cppcheck.results \
         --relative-paths="$CI_PROJECT_DIR" \
         --inline-suppr \
         --suppressions-list=util/suppressions.txt
```


Cppcheck report - BIND 9 (1248f05a) Cppcheck Report:

Defect summary:		Line	Id	CWE	Severity	Message
☒ Toggle all						
Show # Defect ID						
☒	28	nullptrRedundantCheck	101	unknownMacro	error	There is an unknown macro here somewhere. Configuration is required. If DEPRECATEDIN_0_9_8 is a macro then please configure it.
☒	5	nullptrArithmeticRedundantCheck	1354	lib/dns/client.c	warning	Either the condition 'rctx==NULL' is redundant or there is possible null pointer dereference: rctx.
☒	1	unknownMacro	1361	lib/dns/client.c	warning	Either the condition 'rctx==NULL' is redundant or there is possible null pointer dereference: rctx.
34 total						
Statistics						
		1759	lib/dns/tests/rbt_serialize_test.c	476	warning	Either the condition 'ctx==NULL' is redundant or there is possible null pointer dereference: ctx.
		142	lib/dns/zoneverify.c	476	warning	Either the condition 'data!=NULL' is redundant or there is possible null pointer dereference: data.
		150	lib/dns/zoneverify.c	476	warning	Either the condition 'data!=NULL' is redundant or there is possible null pointer dereference: data.
		151	lib/dns/zoneverify.c	476	warning	Either the condition 'data!=NULL' is redundant or there is possible null pointer dereference: data.
		160	lib/dns/zoneverify.c	476	warning	Either the condition 'data!=NULL' is redundant or there is possible null pointer dereference: data.
		405	lib/isc/mem.c	682	warning	Either the condition 'base!=NULL' is redundant or there is pointer arithmetic with NULL pointer.
		406	lib/isc/mem.c	682	warning	Either the condition 'base!=NULL' is redundant or there is pointer arithmetic with NULL pointer.
		1130	lib/isc/mem.c	682	warning	Either the condition 'first==NULL' is redundant or there is pointer arithmetic with NULL pointer.
		1135	lib/isc/mem.c	476	warning	Either the condition 'first==NULL' is redundant or there is possible null pointer dereference: e.
		393	lib/isc/mem.c	682	warning	Either the condition 'tmp!=NULL' is redundant or there is pointer arithmetic with NULL pointer.
		405	lib/isc/mem.c	682	warning	Either the condition 'tmp!=NULL' is redundant or there is pointer arithmetic with NULL pointer.
		1554	lib/isc/mem.c	476	warning	Either the condition 'mpctx!=NULL' is redundant or there is possible null pointer dereference: mpctx.
		1555	lib/isc/mem.c	476	warning	Either the condition 'mpctx!=NULL' is redundant or there is possible null pointer dereference: mpctx.
		1556	lib/isc/mem.c	476	warning	Either the condition 'mpctx!=NULL' is redundant or there is possible null pointer dereference: mpctx.
		1557	lib/isc/mem.c	476	warning	Either the condition 'mpctx!=NULL' is redundant or there is possible null pointer dereference: mpctx.
		1564	lib/isc/mem.c	476	warning	Either the condition 'mpctx!=NULL' is redundant or there is possible null pointer dereference: mpctx.
		1565	lib/isc/mem.c	476	warning	Either the condition 'mpctx!=NULL' is redundant or there is possible null pointer dereference: mpctx.
		1566	lib/isc/mem.c	476	warning	Either the condition 'mpctx!=NULL' is redundant or there is possible null pointer dereference: mpctx.
		1567	lib/isc/mem.c	476	warning	Either the condition 'mpctx!=NULL' is redundant or there is possible null pointer dereference: mpctx.
		1568	lib/isc/mem.c	476	warning	Either the condition 'mpctx!=NULL' is redundant or there is possible null pointer dereference: mpctx.
		1569	lib/isc/mem.c	476	warning	Either the condition 'mpctx!=NULL' is redundant or there is possible null pointer dereference: mpctx.
		1570	lib/isc/mem.c	476	warning	Either the condition 'mpctx!=NULL' is redundant or there is possible null pointer dereference: mpctx.
		1572	lib/isc/mem.c	476	warning	Either the condition 'mpctx!=NULL' is redundant or there is possible null pointer dereference: mpctx.
		1574	lib/isc/mem.c	476	warning	Either the condition 'mpctx!=NULL' is redundant or there is possible null pointer dereference: mpctx.
		494	lib/isc/pk11.c	476	warning	Either the condition 'token!=NULL' is redundant or there is possible null pointer dereference: token.
		495	lib/isc/pk11.c	476	warning	Either the condition 'token!=NULL' is redundant or there is possible null pointer dereference: token.
		283	lib/isc/timer.c	476	warning	Either the condition '(manager)!=NULL' is redundant or there is possible null pointer dereference: manager.
		367	lib/isc/timer.c	476	warning	Either the condition '(timer)!=NULL' is redundant or there is possible null pointer dereference: timer.
		667	lib/isc/timer.c	476	warning	Either the condition '(timer)!=NULL' is redundant or there is possible null pointer dereference: timer.


```

isc_result_t
isc_timer_create(isc_ttermgr_t *manager0, isc_timertype_t type,
                 const isc_time_t *expires, const isc_interval_t *interval,
                 isc_task_t *task, isc_taskaction_t action, void *arg,
                 isc_timer_t **timerp)
{
    isc__ttermgr_t *manager = (isc__ttermgr_t *)manager0; <--- Assignment 'manager=(struct isc__ttermgr*)manager0', assigned value is 0
    isc_timer_t *timer;
    isc_result_t result;
    isc_time_t now;

    /*
     * Create a new 'type' timer managed by 'manager'. The timers
     * parameters are specified by 'expires' and 'interval'. Events
     * will be posted to 'task' and when dispatched 'action' will be
     * called with 'arg' as the arg value. The new timer is returned
     * in 'timerp'.
     */

    REQUIRE(VALID_MANAGER(manager)); <--- Assuming that condition '(manager)!=NULL' is not redundant
    REQUIRE(task != NULL);
    REQUIRE(action != NULL);
    if (expires == NULL)
        expires = isc_time_epoch;
    if (interval == NULL)
        interval = isc_interval_zero;
    REQUIRE(type == isc_timertype_inactive ||
            !(isc_time_isepoch(expires) && isc_interval_iszero(interval)));
    REQUIRE(timerp != NULL && *timerp == NULL);
    REQUIRE(type != isc_timertype_limited ||
            !(isc_time_isepoch(expires) || isc_interval_iszero(interval)));

    /*
     * Get current time.
     */
    if (type != isc_timertype_inactive) {
        TIME_NOW(&now);
    } else {
        /*
         * We don't have to do this, but it keeps the compiler from
         * complaining about "now" possibly being used without being
         * set, even though it will never actually happen.
         */
        isc_time_settoepoch(&now);
    }

    timer = isc_mem_get(manager->mctx, sizeof(*timer)); <--- Null pointer dereference

```

Coccinelle

- Program matching and transformation engine
- Semantic patching

Coccinelle usage

```
$ spatch --sp-file=unreachable.spatch --use-gitgrep --dir  
. --very-quiet --include-headers  
223 files match
```


Coccinelle example

```
$ cat /tmp/cocci/unreachable.spatch
```

```
@@
```

```
@@
```

```
    INSIST(0);
```

```
+    ISC_UNREACHABLE();
```

```
    ... when  $\neq$  ISC_UNREACHABLE();
```


Coccinelle - more complicated example

```
@@
statement S;
expression V;
@@

V = isc_mem_get(...);
- if (V == NULL) S

@@
type T;
statement S;
expression V;
@@

V = (T *)isc_mem_get(...);
- if (V == NULL) S

@@
statement S;
expression V;
@@

if (V == NULL) V = isc_mem_get(...);
- if (V == NULL) S
```

```
@@
statement S1, S2;
expression V;
@@

V = isc_mem_get(...);
- if (V == NULL) S1 else { S2 }
+ S2

@@
type T;
expression V, E1, E2;
@@

- V = (T)isc_mem_get(E1, E2);
+ V = isc_mem_get(E1, E2);
```

Coccinelle - more examples

- http://coccinelle.lip6.fr/impact_linux.php
- <https://gitlab.isc.org/isc-projects/bind9/tree/master/cocci>
- <http://coccinellery.org>

Why is multithreading hard?

- Who understands memory ordering?
 - On a platform with weak memory consistency?
- And what about memory barriers?
- What's lock ellision?

Thread Sanitizer

- Runtime Analyzer
- Requires a good test suite to run
- Or production environment :trollface:
- Checks for:
 - Data races (unprotected memory access)
 - Mutex ordering
 - ...other subtle errors

ThreadSanitizer - usage

```
$ CFLAGS=,-O2 -fno-omit-frame-pointer -fsanitize=thread \
LDFLAGS="-fsanitize=thread" \
./configure --enable-pthread-rwlock
$ make
$ export TSAN_OPTIONS="second_deadlock_stack=1
history_size=7 log_exe_name=true log_path=tsan exitcode=0"
$ make check # have extensive test suite or run the binary
```

Thread Sanitizer: data race (1)

WARNING: ThreadSanitizer: data race (pid=29181)

Write of size 8 at 0x7b90000a0010 by thread T16 (mutexes: write M562522896233146464):

#0 nmhandle_free /home/ondrej/Projects/bind9/lib/isc/netmgr/netmgr.c:1027 (libisc.so.1504+0x3e3b7)

#1 nmhandle_deactivate /home/ondrej/Projects/bind9/lib/isc/netmgr/netmgr.c:1057 (libisc.so.1504+0x3e59f)

#2 **isc_nmhandle_unref** /home/ondrej/Projects/bind9/lib/isc/netmgr/netmgr.c:1108 (libisc.so.1504+0x409f0)

#3 fetch_callback /home/ondrej/Projects/bind9/lib/ns/query.c:5680 (libns.so.1502+0x46b71)

#4 dispatch /home/ondrej/Projects/bind9/lib/isc/task.c:1134 (libisc.so.1504+0x56f36)

#5 run /home/ondrej/Projects/bind9/lib/isc/task.c:1319 (libisc.so.1504+0x56f36)

#6 <null> <null> (libtsan.so.0+0x29b3d)

Previous read of size 4 at 0x7b90000a0010 by thread T1:

#0 **isc_nmhandle_unref** /home/ondrej/Projects/bind9/lib/isc/netmgr/netmgr.c:1067 (libisc.so.1504+0x4091d)

#1 isc__nm_uvreq_put /home/ondrej/Projects/bind9/lib/isc/netmgr/netmgr.c:1214 (libisc.so.1504+0x41bef)

#2 udp_send_cb /home/ondrej/Projects/bind9/lib/isc/netmgr/udp.c:439 (libisc.so.1504+0x46c1d)

#3 <null> <null> (libuv.so.1+0x1d283)

#4 <null> <null> (libtsan.so.0+0x29b3d)

[...]

SUMMARY: ThreadSanitizer: data race lib/isc/netmgr/netmgr.c:1027 in nmhandle_free

isc_nmhandle_unref()

```
void
isc_nmhandle_unref(isc_nmhandle_t *handle) {
    isc_nmsocket_t *sock = NULL, *tmp = NULL;

    REQUIRE(VALID_NMHANDLE(handle));

    if (isc_refcount_decrement(&handle→references) > 1) {
        return;
    }
    [...]
    nmhandle_deactivate(sock, handle); /* calls nmhandle_free() */
    [...]
}
```


nmhandle_free()

```
static void
nmhandle_free(isc_nmsocket_t *sock, isc_nmhandle_t *handle) {
    size_t extra = sock→extrahandlesize;
    if (handle→dofree ≠ NULL) {
        handle→dofree(handle→opaque);
    }

    *handle = (isc_nmhandle_t) {
        .magic = 0
    };

    isc_mem_put(sock→mgr→mctx, handle, sizeof(isc_nmhandle_t) + extra);
}
```

Looks innocuous, right?

nmhandle_free: the fix

```
diff --git a/lib/isc/netmgr/netmgr.c b/lib/isc/netmgr/netmgr.c
index 8317bf27a7..7618ff09a9 100644
--- a/lib/isc/netmgr/netmgr.c
+++ b/lib/isc/netmgr/netmgr.c
@@ -1022,6 +1022,8 @@ static void
nmhandle_free(isc_nmsocket_t *sock, isc_nmhandle_t *handle) {
    size_t extra = sock->extrahandlesize;

+   isc_refcount_destroy(&handle->references);
+
    if (handle->dofree != NULL) {
        handle->dofree(handle->opaque);
    }
```

isc_refcount API

```
#define isc_refcount_current(target) \  
    atomic_load_explicit(target, memory_order_acquire)  
  
#define isc_refcount_destroy(target) \  
    ISC_REQUIRE(isc_refcount_current(target) == 0)  
  
#define isc_refcount_decrement(target) \  
    atomic_fetch_sub_explicit(target, 1, memory_order_release)
```

Release-Acquire ordering

- If an atomic store in thread A is tagged `memory_order_release` and an atomic load in thread B from the same variable is tagged `memory_order_acquire`, all memory writes (non-atomic and relaxed atomic) that happened-before the atomic store from the point of view of thread A, become visible side-effects in thread B. That is, once the atomic load is completed, thread B is guaranteed to see everything thread A wrote to memory.
- **The synchronization is established only between the threads releasing and acquiring the same atomic variable.** Other threads can see different order of memory accesses than either or both of the synchronized threads.



Data race on ppc64le

- BIND 9 Issue #1428
- Most probably caused by BIND 9 rwlock using stdatomic.

Proposed patch (1/2)

```
--- a/lib/isc/rwlock.c
+++ b/lib/isc/rwlock.c
@@ -434,7 +434,7 @@ isc_rwlock_trylock(isc_rwlock_t *rwl, isc_rwlocktype_t
type) {
    #if defined(ISC_RWLOCK_USESTDATOMIC)
        cntflag = atomic_fetch_add_explicit(&rwl->cnt_and_flag,
                                            READER_INCR,
-                                           memory_order_relaxed);
+                                           memory_order_acquire);
    #else
        cntflag = isc_atomic_xadd(&rwl->cnt_and_flag, READER_INCR);
    #endif
```


Proposed patch (2/2)

```
--- a/lib/isc/rwlock.c
+++ b/lib/isc/rwlock.c
@@ -470,7 +470,7 @@ isc_rwlock_trylock(isc_rwlock_t *rwl,
isc_rwlocktype_t type) {
    int_fast32_t zero = 0;
    if (!atomic_compare_exchange_strong_explicit
        (&rwl->cnt_and_flag, &zero, WRITER_ACTIVE,
-         memory_order_relaxed, memory_order_relaxed))
+         memory_order_acquire, memory_order_relaxed))
        return (ISC_R_LOCKBUSY);

    #else

    cntflag = isc_atomic_cmpxchg(&rwl->cnt_and_flag, 0,
```




...no one else has this socket

```
diff --git a/lib/isc/unix/socket.c b/lib/isc/unix/socket.c
index 2c97e6e697..fc93d369b5 100644
--- a/lib/isc/unix/socket.c
+++ b/lib/isc/unix/socket.c
@@ -2596,15 +2596,16 @@ isc_socket_open(isc_socket_t *sock0) {

    REQUIRE(VALID_SOCKET(sock));

-   REQUIRE(isc_refcount_current(&sock->references) == 1);
-   /*
-    * We don't need to retain the lock hereafter, since no one else has
-    * this socket.
-    */
+   LOCK(&sock->lock);
+
+   REQUIRE(isc_refcount_current(&sock->references) ≥ 1);
+   REQUIRE(sock->fd == -1);
+   REQUIRE(sock->threadid == -1);

    result = opensocket(sock->manager, sock, NULL);
+
+   UNLOCK(&sock->lock);
+
    if (result ≠ ISC_R_SUCCESS) {
        sock->fd = -1;
    } else {
```

**...no one else has
this socket**

except when on FreeBSD in a
VMWare on a hardware with lot
of cores



Photo by [Peter Forster](#) on [Unsplash](#)

The mysterious crash in libthr

- Additions to our GitLab CI:
 - FreeBSD in September 2019
 - Windows in September 2019
 - OpenBSD in October 2019
- And there was an intermittent failure in the FreeBSD system test job
 - Crash in the libthr system library

OpenSSL_atexit() concurrency

```
diff --git a/bin/named/server.c b/bin/named/server.c
index 55663d515b..487acdcf08 100644
--- a/bin/named/server.c
+++ b/bin/named/server.c
@@ -10016,7 +10017,15 @@ named_server_destroy(named_server_t **serverp) {
}
```

```
static void
-fatal(const char *msg, isc_result_t result) {
+fatal(named_server_t *server, const char *msg, isc_result_t result) {
+    if (server != NULL) {
+        /*
+         * Prevent races between the OpenSSL on_exit registered
+         * function and any other OpenSSL calls from other tasks
+         * by requesting exclusive access to the task manager.
+         */
+        (void)isc_task_beginexclusive(server->ta
+    }
+    isc_log_write(named_g_lctx, NAMED_LOGCATEGORY_GENERAL,
+                  NAMED_LOGMODULE_SERVER, ISC_LOG_CRITICAL,
+                  "%s: %s", msg, isc_result_totext(result));
```

As of version 1.1.0 OpenSSL will automatically allocate all resources that it needs so no explicit initialisation is required. Similarly it will also automatically deinitialise as required.

–The OpenSSL Project Authors

ThreadSanitizer: lock-order-inversion

```
WARNING: ThreadSanitizer: lock-order-inversion (potential deadlock) (pid=21211)  
  Cycle in lock order graph: M1110 (0x7b740000000008) ⇒ M1728 (0x7b4c0000001d0)  
⇒ M1110
```

```
Mutex M1728 acquired here while holding mutex M1110 in main thread:
```

```
[...]
```

```
Mutex M1110 previously acquired by the same thread here:
```

```
[...]
```

```
Mutex M1110 acquired here while holding mutex M1728 in main thread:
```

```
[...]
```

```
Mutex M1728 previously acquired by the same thread here:
```

```
[...]
```

```
SUMMARY: ThreadSanitizer: lock-order-inversion (potential deadlock) (/usr/lib/  
x86_64-linux-gnu/libtsan.so.0+0x3d62b) in pthread_mutex_lock
```


Convert simple variables to stdatomic

```
@@ -1521,10 +1521,6 @@ fcount_incr(fetchctx_t *fctx, bool force) {
-     LOCK(&fctx→res→lock);
-     spill = fctx→res→zspill;
-     UNLOCK(&fctx→res→lock);
@@ -1549,6 +1545,7 @@ fcount_incr(fetchctx_t *fctx, bool force) {
    } else {
+         uint_fast32_t spill = atomic_load_acquire(&fctx→res→zspill);
+         if (!force && spill ≠ 0 && counter→count ≥ spill) {
@@ -9969,7 +9966,7 @@ dns_resolver_create(dns_view_t *view,
-     res→zspill = 0;
+     atomic_init(&res→zspill, 0);
@@ -11249,9 +11246,7 @@ dns_resolver_setfetchesperzone(dns_resolver_t *resolver, uint32_t
clients)
-     LOCK(&resolver→lock);
-     resolver→zspill = clients;
-     UNLOCK(&resolver→lock);
+     atomic_store_release(&resolver→zspill, clients);
    }
```

ThreadSanitizer
Are we there yet?

Not yet!

```
2 x data race in free
16 x data race in epoll_ctl
7 x data race in memset
1 x data race lib/dns/message.c:397:13 in msginittsig
4 x data race lib/dns/rbtdb.c:1545:6 in mark_header_stale
1 x data race lib/dns/rbtdb.c:1557:21 in mark_header_stale
1 x data race lib/dns/rbtdb.c:4336:7 in check_stale_header
92 x lock-order-inversion (potential deadlock) in pthread_mutex_lock
88 x lock-order-inversion (potential deadlock) in pthread_rwlock_rdlock
5 x lock-order-inversion (potential deadlock) in pthread_rwlock_wrlock
```


A lot of duplicates here, but still...

```
2 x data race in free
16 x data race in epoll_ctl
7 x data race in memset
1 x data race lib/dns/message.c:397:13 in msginittsig
4 x data race lib/dns/rbtdb.c:1545:6 in mark_header_stale
1 x data race lib/dns/rbtdb.c:1557:21 in mark_header_stale
1 x data race lib/dns/rbtdb.c:4336:7 in check_stale_header
92 x lock-order-inversion (potential deadlock) in pthread_mutex_lock
88 x lock-order-inversion (potential deadlock) in pthread_rwlock_rdlock
5 x lock-order-inversion (potential deadlock) in pthread_rwlock_wrlock
```

10 unique warnings
217 unique paths



Photo by [Aaron Burden](#) on [Unsplash](#)

Thank you for listening!

Questions?

Ondřej Surý <ondrej@isc.org>