C++test 静态分析

—— 来自某跨国网通设备大厂真实客户案例

PARASOFT

一、问题重现

下面的描述来自 Parasoft 的客户某跨国网通设备大厂的邮件内容，客户在邮件中描述了一段存在 Bug 的代码。以下是邮件内容（注：为防止泄漏客户代码信息，对代码做过轻微改动）：

“

As talked last night, we have solved one big code bug; here I list it here for your information.

This code cause system running in an endless loop, that cause system reboot due to watch dog timer timeout.

```
void ETN_xdd_algo::stop_running(T_global_TCID tcid)
{
    if ( pinstances == NULL) {
        return;
    }
    LE_simple_list<ETN_xdd_algo *>::iterator it =
pinstances->begin();
    for (; it != pinstances->end(); ) {
        if (tcid == (*it)->get_tcid()) {
            delete *it;
            break;
        }
    }
    return;
}
```

”

对上述错误代码做简单分析，你将很容易发现问题所在，原因简单甚至弱智，因为 for 循环的循环因子 it 没有进行自加运算。这导致只要进入该 for 循环，必将陷入死循环。

以上是从代码角度进行分析，得出死循环结论。 而当代码在实际项目中运行时（上述错误代码的项目，实际运行在客户所生产的电信设备上），所产生的真实后果则更为严重。

上海市打浦路 1 号金玉兰广场西楼 902 室
Rm. 902, Golden Magnolia Plaza, W. 1 Dapu Road, Huangpu District 200023 Shanghai, P.R. China
www.parasoft.com

客户在邮件中有简单提及：

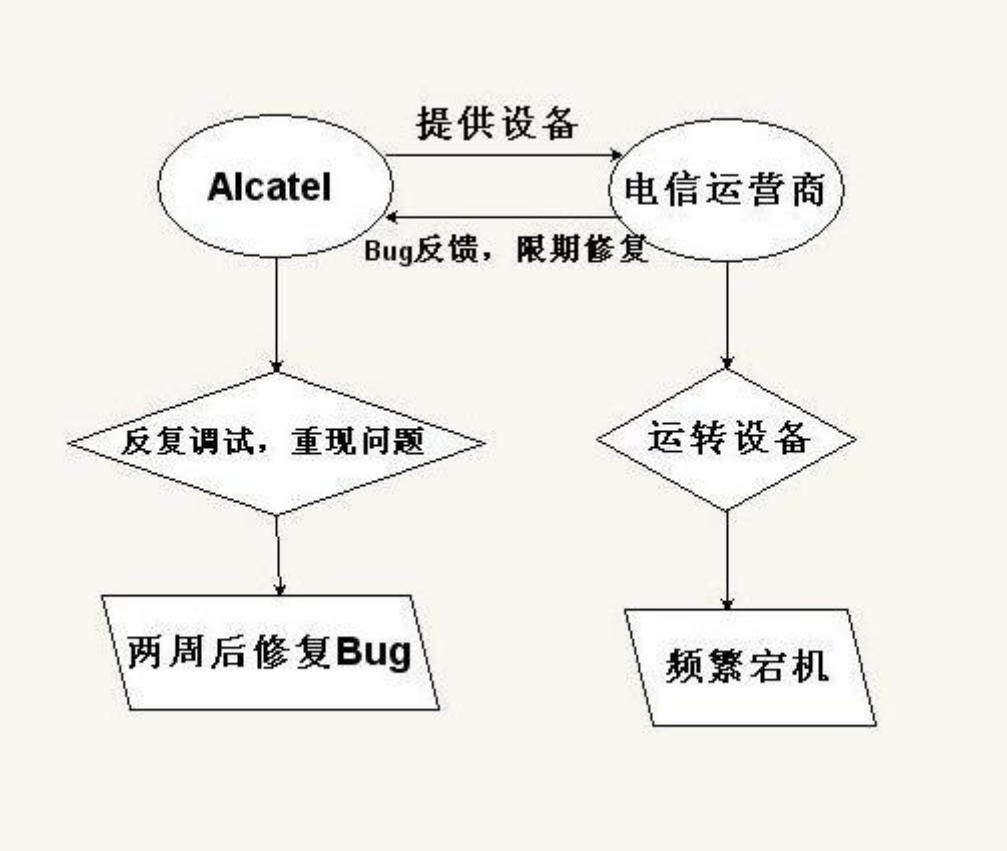
“solved one big code bug”

“This code cause system running in an endless loop, that cause system reboot due to watch dog timer timeout.”

而在我们现场支持客户端过程中，客户曾说明如下：

“由于存在 Bug 的设备在电信运营商的系统上，经常无端死机，导致他们的客户怨声载道。责令他们必须在规定期限内，找到问题根源并修复；”

好了，我们再简单回顾一下整个问题发生的流程，以便我们能更好的重现问题：



作为电信设备供应商的某跨国网通设备大厂，所销售的设备正运行在运营商系统上，而运营商发现该设备经常无端死机，于是要求某跨国网通设备大厂必须在短期内找到问题根源并修复。由于不能现场调试，某跨国网通设备大厂的开发人员只能在本地反复诊断，反复模

上海市打浦路 1 号金玉兰广场西楼 902 室
Rm. 902, Golden Magnolia Plaza, W. 1 Dapu Road, Huangpu District 200023 Shanghai, P.R. China
www.parasoft.com

拟，希望重现死机的现象，经过长达一周时间的模拟，他们终于发现问题的根源正是上述那段简单的 for 循环错误。

二、客户背景

某跨国网通设备大厂是国际电信行业的佼佼者，是数十亿美元规模的电信技术领先厂商，拥有数个重要的全球研发中心，拥有技术先进、制造能力达到世界一流水平的生产制造平台，销售服务网络遍及全国和海外 50 多个国家。

三、客户代码

上述错误的代码正来自某跨国网通设备大厂核心设备的软件中。述及该项目的历史，也足让你瞠目结舌。总项目全部基础性代码超过 1000 万行，最早的代码可追踪到 90 年代。客户所测项目隶属总项目重要模块之一，代码量超过 100 万行。

四、问题影响

由于实际项目已经运行于电信设备上，而该 Bug 只会在极端情况下才会发生。可想而知，这样的问题从他们客户那里反馈回来，然后该项目主要负责人开始安排人手从 100 万行代码中查找该问题，何异于大海捞针。无赖之下，客户只有不停模拟各种输入参数，模拟各种可能发生的情况。经数日苦战，客户终于重现该问题。然后顺藤摸瓜，揪出以上“罪魁祸首”的源码 – “该死的 For 循环!”

更为重要的影响在于运营商对于某跨国网通设备大厂产品质量的认定，这样的问题对于任何客户而言都是不希望看到的，因为这会影响到实际产品的运行，进而消费者对于产品的使用。

五、解决之道

在软件开发中，我们极不情愿将 Bug 呈现到客户面前，但当问题真正产生时，我们需要去反思为什么如此简单的 for 循环所导致的问题造成这么大影响。

- 产品没有经过测试吗？有测试，但往往是功能性和系统性测试。

上海市打浦路 1 号金玉兰广场西楼 902 室
Rm. 902, Golden Magnolia Plaza, W. 1 Dapu Road, Huangpu District 200023 Shanghai, P.R. China
www.parasoft.com

- 没有代码规范标准吗？有标准，但往往停留于纸面而不得执行。
- 项目初期即介入测试吗？

……

以上一些问题是保障软件产品高质量的一些因素，对于这些，C++test 的静态规则检查正好完美的满足这些需求：

- 在代码级别测试上，C++test 的静态分析能建立有效代码规范标准，使得测试不仅仅局限于功能性和系统性测试；
- 在代码规范标准上，C++test 内建近 1600 条代码规范，同时提供自定义代码规范工具 RuleWizard；
- C++test 的静态分析，Bug Detective 功能，不需要整个项目完成后测试，可在编写代码的第一行开始，即进行测试。尽早消除代码隐患。

对于上文中某跨国网通设备大厂所出现的 Bug，只需要通过 C++test 一条简单代码规范，即可防范于未然。

以下规则来自 C++test 静态编码规则：

“Null initialize or increment expressions in for loops will not be used; a while loop will be used instead [CODSTA-49-3]”

六、客户反馈

从某跨国网通设备大厂反馈的情况，他们非常信赖 C++test 静态分析，已经将 C++test 的静态分析作为测试的标准流程，每日进行测试。因为通过 C++test 的代码规范，只需数分钟即可预防该问题发生。而该问题从发现到修复前后花费 2 周时间，更重要的是影响客户对于他们产品的信心。

上海市打浦路 1 号金玉兰广场西楼 902 室
Rm. 902, Golden Magnolia Plaza, W. 1 Dapu Road, Huangpu District 200023 Shanghai, P.R. China
www.parasoft.com