

软件 Demo 源码检查报告

版本号: 0.1

发布日期: 2020-09-30





版本历史

| 版本号 | 日期 | 制/修订人 | 内容描述 |
|-----|-----------|-------|--------|
| 0.5 | 2020.9.28 | 汤健雄 | 修改文档名称 |
| 0.1 | 2020.9.19 | 汤健雄 | 建立文档 |







目 录

| 1 | 前言 1.1 文档简介 | 1 1 1 |
|---|--|-----------------------|
| 2 | 工具说明 2.1 CPPcheck | 2 2 2 2 |
| 3 | check 静态代码检查 3.1 公共模块 3.2 服务端 A 3.3 客户端 B 3.4 客户端 C 3.5 检查结果 | 3 3 4 4 4 |
| | 3.5 检查结果 | 5 6 |
| | | |



前言

1.1 文档简介

此文档为 p166 软件项目的静态代码审核报告。

1.2 目标读者

P166 项目软件测试人员和项目验收人员。

1.3 文档目的

使用相关的静态代码检查工具对软件源代码进行检查, 检测代码的一些隐藏错误和缺陷,提高软 件的可靠性并节省软件开发和测试的成本。





1具说明

静态代码检测工具种类很多,本文档用到的主要有以下工具。

2.1 CPPcheck

Cppcheck 是一种 C/C++ 代码缺陷静态检查工具,不同于 C/C++ 编译器及其它分析工具, Cppcheck 只检查编译器检查不出来的 bug,不检查语法错误,作为编译器的一种补充检查, cppcheck 对产品的源代码执行严格的逻辑检查。执行的检查包括: MINER

- 自动变量检查
- 数组的边界检查
- class 类检查
- 过期的函数,废弃函数调用检查
- 异常内存使用,释放检查
- 内存泄漏检查,主要是通过内存引用指针
- 操作系统资源释放检查,中断,文件描述符等
- 异常 STL 函数使用检查
- 代码格式错误,以及性能因素检查

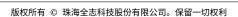
2.2 gcov

gcov 是测试代码覆盖率的工具,可通过命令行方式进行操作,可以帮助开发者优化代码。主要执 行的检查包括

- 每一行代码的执行频率
- 实际上哪些代码确实被执行了
- 每一段代码的执行时间

2.3 checkpatch.pl

checkpatch 是公司在仓库集成的 patch 检查工具进行格式检查,每次发生变更之后需要达到提 交的规范才能进行代码的合并。





B check 静态代码检查

静态代码检查环节采用 cppcheck 工具分别对源码的公共模块、内核模块和应用模块进行了相应的检查。主要发现的问题如下:

3.1 公共模块

```
[src/codec.c:129]: (style) The scope of the variable 'c' can be reduced.
[src/codec.c:184]: (style) The scope of the variable 'c' can be reduced.
[src/codec.c:133]: (style) Checking if unsigned variable 'inlen' is less than zero.
[src/codec.c:187]: (style) Checking if unsigned variable 'inlen' is less than zero.
[src/hash.c:190]: (style) Variable 'str' is assigned a value that is never used.
[src/connect.c:6]: (style) The function 'get_id' is never used.
[src/hash.c:305]: (style) The function 'hash_str' is never used.
```

3.2 服务端 A

```
[src/server_a.c:253]: (style) The scope of the variable 'find' can be reduced.
[src/server_a.c:260]: (style) The scope of the variable 'res' can be reduced.
[src/server_a.c:16]: (style) Variable 'thrd_num' is assigned a value that is never used.
[src/server_a.c:32]: (style) Variable 'thrd_num' is assigned a value that is never used.
[src/server_a.c:111]: (style) Variable 'ret' is assigned a value that is never used.
[src/server_a.c:194]: (style) Variable 'num1' is assigned a value that is never used.
[src/server_a.c:205]: (style) Variable 'num' is assigned a value that is never used.
[src/server_a.c:252]: (style) Unused variable: len
[src/server_a.c:257]: (style) Unused variable: buf
[src/server_a.c:258]: (style) Unused variable: buf_hash
[src/server_a.c:262]: (style) Unused variable: msgtype
[src/server_a.c:14]: (style) The function 'thread_recv_messagel' is never used.
```



3.3 客户端 B

[src/client_b.c:11]: (style) Variable 'thrd_num' is assigned a value that is never used.
[src/client_b.c:121]: (error) fflush() called on input stream 'stdin' results in undefined behaviour.

[src/client_b.c:119]: (portability) scanf without field width limits can crash with huge input data on some versions of libc.

3.4 客户端 C

[src/client_c.c:11]: (style) Variable 'thrd_num' is assigned a value that is never used.
[src/client_c.c:116]: (error) fflush() called on input stream 'stdin' results in undefined behaviour.

[src/client_c.c:114]: (portability) scanf without field width limits can crash with huge input data on some versions of libc.

MIN

3.5 检查结果

各个源码文件均存在或多或少的 style 类型缺陷,主要有以下几个方面

- 变量声明未使用
- 变量值与类型不符
- 变量范围过大

客户端 bc 源文件中出现了 error 类型的缺陷:主要是因为 fflush 函数的使用方法不当。 客户端 bc 源文件中出现了 portability 类型的缺陷:主要原因是 scanf 输入没有字段宽度限制。



4

缺陷统计

表 4-1: cppcheck 静态代码检查缺陷统计

| 缺陷类型 | 所属文件 | 数量 | 解决数量 | 遗留缺陷 |
|-------------|---------------|----|------|------|
| style | server_a.c | 11 | 11 | 0 |
| | client_b.c | 1 | 1 | 0 |
| | $client_c.c$ | 1 | 1 | 0 |
| | codec.c | 4 | 4 | 0 |
| | hash.c | 2 | 2 | 0 |
| | connect.c | 1 | 1 | 0 |
| error | client_b.c | 1 | 1 | 0 |
| | $client_c.c$ | 1 | 1 | 0 |
| portability | client_b.c | 1 | 1 | 0 |
| | client_c.c | 1 | 1 | 0 |
| | | | | |



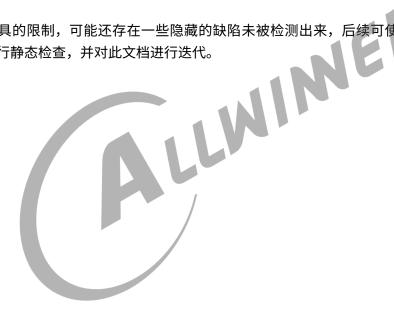


总结

本文档对软件源码的静态审核结果进行了记录,主要是采用 cppchek 工具对源码进行了自动变量 检查、数组的边界检查、class 类检查、废弃函数调用检查、异常内存使用,释放检查内存泄漏检 查等等,根据检测结果对相应的缺陷进行了修改。产生缺陷的主要原因有几点:

- 函数定义之后未使用
- 用户输入未限定边界
- 变量范围过大
- 公共接口函数使用不当

由于检查工具的限制,可能还存在一些隐藏的缺陷未被检测出来,后续可使用其他工具对代码的 其他方面进行静态检查,并对此文档进行迭代。





著作权声明

版权所有 © 2020 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护,其著作权由珠海全志科技股份有限公司("全志")拥有并保留 一切权利。

本文档是全志的原创作品和版权财产,未经全志书面许可,任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部,且不得以任何形式传播。

商标声明



举)均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标,产品名称,和服务名称,均由其各自所有人拥有。

免责声明



本文档作为使用指导仅供参考。由于产品版本升级或其他原因,本文档内容有可能修改,如有变更,恕不另行通知。全志尽全力在本文档中提供准确的信息,但并不确保内容完全没有错误,因使用本文档而发生损害(包括但不限于间接的、偶然的、特殊的损失)或发生侵犯第三方权利事件,全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中,可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税(专利税)。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。