# Static analysis for C++

Figure out the bugs before compiling your code.

SCC

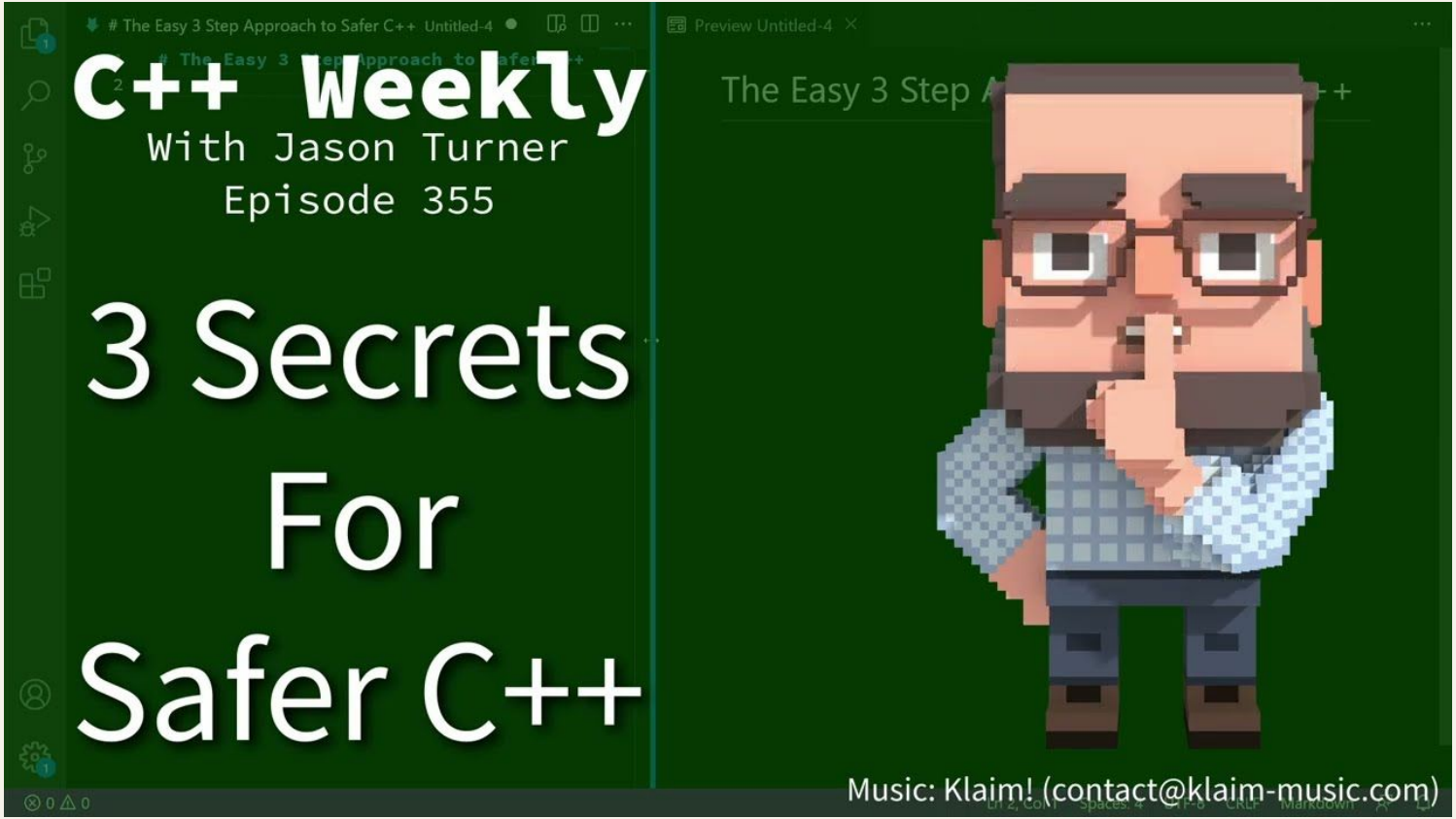
TEAMT5
杜 浦 數 位 安 全
Persistent Cyber Threat Hunters

# Outline

- Enable warning flags (treating warnings as errors)
- Build both `Debug` and `Release` versions
- Msvc /analyze , clang static analyzer
- cppcheck
- clang-tidy

# The C++weekly

https://www.youtube.com/watch?v=dSYFm65KcYo



# cppcon

TEAMT5
杜 浦 數 位 安 全

https://www.youtube.com/watch?v=sn1Vg8A_MPU

https://www.youtube.com/watch?v=4fB7YcyofrE

https://www.youtube.com/watch?v=rKlHvAw1z50

# Warning flags

TEAMT5
杜 浦 數 位 安 全

-Wall

-Werror

-Wextra

(-Wmost)

/Wall

/WX (The linker also has a /WX option.)

/wLnnnn …

/Wp64

# Warning options in MSVC

TEAMT5 杜浦數位安全

Well, I couldn’t find any pattern in those digits.

But I could write a conversion tool. (?)

| | |
|---|---|
| **/W3** | **/W3** displays level 1, level 2, and level 3 (production quality) warnings. **/W3** is the default setting in the IDE. |
| **/W4** | **/W4** displays level 1, level 2, and level 3 warnings, and all level 4 (informational) warnings that aren't off by default. We recommend that you use this option to provide lint-like warnings. For a new project, it may be best to use **/W4** in all compilations. This option helps ensure the fewest possible hard-to-find code defects. |
| **/Wall** | Displays all warnings displayed by **/W4** and all other warnings that **/W4** doesn't include—for example, warnings that are off by default. For more information, see Compiler warnings that are off by default. |

`-Wextra`

This enables some extra warning flags that are not enabled by `-Wall`. (This option used to be called `-W`. The older name is still supported, but the newer name is more descriptive.)

```
-Wclobbered
-Wcast-function-type
-Wdeprecated-copy (C++ only)
-Wempty-body
-Wenum-conversion (C only)
-Wignored-qualifiers
-Wimplicit-fallthrough=3
-Wmissing-field-initializers
-Wmissing-parameter-type (C only)
-Wold-style-declaration (C only)
-Woverride-init
-Wsign-compare (C only)
-Wstring-compare
-Wredundant-move (only for C++)
-Wtype-limits
-Wuninitialized
-Wshift-negative-value (in C++11 to C++17 and in C99 and newer)
-Wunused-parameter (only with -Wunused or -Wall)
-Wunused-but-set-parameter (only with -Wunused or -Wall)
```

The option `-Wextra` also prints warning messages for the following cases:

- A pointer is compared against integer zero with `<`, `<=`, `>`, or `>=`.
- (C++ only) An enumerator and a non-enumerator both appear in a conditional expression.
- (C++ only) Ambiguous virtual bases.
- (C++ only) Subscripting an array that has been declared `register`.
- (C++ only) Taking the address of a variable that has been declared `register`.
- (C++ only) A base class is not initialized in the copy constructor of a derived class.

TEAMT5
杜 浦 數 位 安 全

## -Wmost

Some of the diagnostics controlled by this flag are enabled by default.

Controls **-Warray-parameter**, **-Wbool-operation**, **-Wcast-of-sel-type**, **-Wchar-subscripts**, **-Wcomment**, **-Wdelete-non-virtual-dtor**, **-Wextern-c-compat**, **-Wfor-loop-analysis**, **-Wformat**, **-Wframe-address**, **-Wimplicit**, **-Winfinite-recursion**, **-Wint-in-bool-context**, **-Wmismatched-tags**, **-Wmissing-braces**, **-Wmove**, **-Wmultichar**, **-Wobjc-designated-initializers**, **-Wobjc-flexible-array**, **-Wobjc-missing-super-calls**, **-Woverloaded-virtual**, **-Wprivate-extern**, **-Wrange-loop-construct**, **-Wreorder**, **-Wreturn-type**, **-Wself-assign**, **-Wself-move**, **-Wsizeof-array-argument**, **-Wsizeof-array-decay**, **-Wstring-plus-int**, **-Wtautological-compare**, **-Wtrigraphs**, **-Wuninitialized**, **-Wunknown-pragmas**, **-Wunused**, **-Wuser-defined-warnings**, **-Wvolatile-register-var**.

## -Wall

Some of the diagnostics controlled by this flag are enabled by default.

Controls **-Wmisleading-indentation**, **-Wmost**, **-Wparentheses**, **-Wswitch**, **-Wswitch-bool**.

## -Wextra ¶

Some of the diagnostics controlled by this flag are enabled by default.

Also controls **-Wdeprecated-copy**, **-Wempty-init-stmt**, **-Wfuse-ld-path**, **-Wignored-qualifiers**, **-Winitializer-overrides**, **-Wmissing-field-initializers**, **-Wmissing-method-return-type**, **-Wnull-pointer-arithmetic**, **-Wnull-pointer-subtraction**, **-Wsemicolon-before-method-body**, **-Wsign-compare**, **-Wstring-concatenation**, **-Wunused-but-set-parameter**, **-Wunused-parameter**.

# Why different build versions

TEAMT5
杜 浦 數 位 安 全

**Optimization** flags would affect statistical analysis.

- We will also introduce dynamic analysis via compiling flags.

# Different compilers' compilation

TEAMT5 杜浦數位安全

| | Warning/Checking | Compile time optimization | C++17 supportion |
|---|---|---|---|
| MSVC | Lazy | Lazy | Complete 19.34 / Almost 19.14 |
| GCC | Medium | Aggressive | Complete 11/ Almost 7 |
| Clang | Aggressive | Medium | Not Complete Yet / Almost 4 |

Src: https://www.youtube.com/watch?v=4pKtPWcl1Go
https://en.cppreference.com/w/cpp/compiler_support/17

# /analyze in MSVC

Enabled on:

Configuration Properties > Code Analysis > General property page.

| 程式碼 | 說明 |
|---|---|
| C26495 | Variable 'Base::dummy_int' is uninitialized. Always initialize a member variable (type.6). |
| C26482 | Only index into arrays using constant expressions (bounds.2). |
| C26446 | Prefer to use gsl::at() instead of unchecked subscript operator (bounds.4). |
| C26493 | Don't use C-style casts (type.4). |
| C26447 | The function is declared 'noexcept' but calls function 'operator<<()' which may throw exceptions (f.6). |
| C26447 | The function is declared 'noexcept' but calls function 'char_traits<char> >()' which may throw exceptions (f.6). |
| C26447 | The function is declared 'noexcept' but calls function 'operator<<()' which may throw exceptions (f.6). |
| C26447 | The function is declared 'noexcept' but calls function 'operator<<()' which may throw exceptions (f.6). |
| C26447 | The function is declared 'noexcept' but calls function 'char_traits<char> >()' which may throw exceptions (f.6). |

錯誤清單 輸出 呼叫階層

ConsoleApplication2 - Microsoft Visual Studio

檔案(F) 編輯(E) 檢視(V) 專案(P) 建置(B) 偵錯(D) 小組(M) 工具(T) 測試(S) 分析(N) 視窗(W) 說明(H)

Debug x86 本機 Windows 偵錯工具

ConsoleApplication2.cpp

ConsoleApplication2 | Base | Base()

```cpp
#include <memory>
#include <array>
#include <iostream>
#include <utility>

struct Base
{
    Base() noexcept
    {
        for (int i = 0; i < 10; i++)
            dummy_int[i] = i;
        for (auto &i : dummy_ptr)
            i = (int *)1;
        for (auto &i : dummy_bool)
            i = 1;
    };

    ~Base() = default;

    std::array<int, 10> dummy_int;
    std::array<int *, 10> dummy_ptr;
    std::array<bool, 10> dummy_bool;

    void print() const noexcept
    {
        for (const auto &i : dummy_int)
            std::cout << i << " ";
        std::cout << std::endl;

        for (const auto &i : dummy_ptr)
            std::cout << i << " ";
        std::cout << std::endl;

        for (const auto &i : dummy_bool)
            std::cout << i << " ";
```

100 %

錯誤清單

整個方案 | 0 錯誤 | 19 警告 | 0 訊息 | 僅限組建

| 程式碼 | 說明 | 專案 | 檔案 | 行 | 隱藏項目狀態 |
|---|---|---|---|---|---|
| C26495 | Variable 'Base::dummy_int' is uninitialized. Always initialize a member variable (type.6). | ConsoleApplication2 | consoleapplication2.cpp | 8 | 使用中 |
| C26482 | Only index into arrays using constant expressions (bounds.2). | ConsoleApplication2 | consoleapplication2.cpp | 11 | 使用中 |
| C26446 | Prefer to use gsl::at() instead of unchecked subscript operator (bounds.4). | ConsoleApplication2 | consoleapplication2.cpp | 11 | 使用中 |
| C26493 | Don't use C-style casts (type.4). | ConsoleApplication2 | consoleapplication2.cpp | 13 | 使用中 |
| C26447 | The function is declared 'noexcept' but calls function 'operator<<()' which may throw exceptions (f.6). | ConsoleApplication2 | consoleapplication2.cpp | 27 | 使用中 |
| C26447 | The function is declared 'noexcept' but calls function 'char_traits<char> >()' which may throw exceptions (f.6). | ConsoleApplication2 | consoleapplication2.cpp | 27 | 使用中 |
| C26447 | The function is declared 'noexcept' but calls function 'operator<<()' which may throw exceptions (f.6). | ConsoleApplication2 | consoleapplication2.cpp | 28 | 使用中 |
| C26447 | The function is declared 'noexcept' but calls function 'operator<<()' which may throw exceptions (f.6). | ConsoleApplication2 | consoleapplication2.cpp | 31 | 使用中 |
| C26447 | The function is declared 'noexcept' but calls function 'char_traits<char> >()' which may throw exceptions (f.6). | ConsoleApplication2 | consoleapplication2.cpp | 31 | 使用中 |

錯誤清單 輸出 呼叫階層

方案總管

搜尋方案總管 (Ctrl+;)

- 方案 'ConsoleApplication2' (1 專案)
  - ConsoleApplication2
    - 參考
    - 外部相依性
    - 來源檔案
    - 資源檔
    - 標頭檔

就緒 新增至原始檔控制

下午 10:52 2023/1/8

ConsoleApplication2 - Microsoft Visual Studio

檔案(F) 編輯(E) 檢視(V) 專案(P) 建置(B) 偵錯(D) 小組(M) 工具(T) 測試(S) 分析(N) 視窗(W) 說明(H)

Debug x86 本機 Windows 偵錯工具

ConsoleApplication2.cpp

ConsoleApplication2 | Base | Base()

```cpp
#include <memory>
#include <array>
#include <iostream>
#include <utility>

struct Base
{
    Base() noexcept
    {
        for (int i = 0; i < 10; i++)
            dummy_int[i] = i;
        for (auto &i : dummy_ptr)
            i = (int *)1;
        for (auto &i : dummy_bool)
            i = 1;
    };

    ~Base() = default;

    std::array<int, 10> dummy_int;
    std::array<int *, 10> dummy_ptr;
    std::array<bool, 10> dummy_bool;

    void print() const noexcept
    {
        for (const auto &i : dummy_int)
            std::cout << i << " ";
        std::cout << std::endl;

        for (const auto &i : dummy_ptr)
            std::cout << i << " ";
        std::cout << std::endl;

        for (const auto &i : dummy_bool)
            std::cout << i << " ";
```

100 %

輸出

顯示輸出來源(S): 建置

```
1>------ 已開始全部重建: 專案: ConsoleApplication2, 組態: Debug Win32 ------
1>ConsoleApplication2.cpp
1>ConsoleApplication2.vcxproj -> C:\Users\zxc25\source\repos\ConsoleApplication2\Debug\ConsoleApplication2.exe
========== 全部重建: 1 成功、0 失敗、 0 略過 ==========
```

錯誤清單 輸出 呼叫階層

方案總管

搜尋方案總管 (Ctrl+;)

- 方案 'ConsoleApplication2' (1 專案)
  - ConsoleApplication2
    - 參考
    - 外部相依性
    - 來源檔案
    - 資源檔
    - 標頭檔

就緒 新增至原始檔控制

下午 10:54 2023/1/8

ConsoleApplication1 - Microsoft Visual Studio

檔案(F) 編輯(E) 檢視(V) 專案(P) 建置(B) 偵錯(D) 小組(M) 工具(T) 測試(S) 分析(N) 視窗(W) 說明(H)

Debug x86 本機 Windows 偵錯工具

ConsoleApplication1.cpp

ConsoleApplication1 (全域範圍)

```
#include <iostream>
#include <memory>

void foo(const int* const p) {
    if (p != nullptr)
        std::cout << *p << std::endl;
}

int main()
{
    auto ptr = std::make_unique<int>(42);

    foo(ptr.release());

    std::cout << *ptr.get() << std::endl;

    return 0;
}
```

100 %

輸出

顯示輸出來源(S): 建置

```
1>------ 已開始全部重建: 專案: ConsoleApplication1, 組態: Debug Win32 ------
1>ConsoleApplication1.cpp
1>ConsoleApplication1.vcxproj -> c:\users\daven\source\repos\ConsoleApplication1\Debug\ConsoleApplication1.exe
========== 全部重建: 1 成功、0 失敗、 0 略過 ==========
```

錯誤清單 輸出

(名稱)
指定專案名稱。

全部重建成功

新增至原始檔控制

ConsoleApplication1 屬性頁

組態(C): 作用中 (Debug) 平台(P): 作用中 (Win32) 組態管理員(O)...

- 組態屬性
  - 一般
  - 偵錯
  - VC++ 目錄
  - C/C++
    - 一般
    - 最佳化
    - 前置處理器
    - 程式碼產生
    - 語言
    - 先行編譯的標頭檔
    - 輸出檔
    - 瀏覽資訊
    - 進階
    - 所有選項
    - 命令列
  - 連結器
  - 資訊清單工具
  - XML 文件產生器
  - 瀏覽資訊
  - 建置事件
  - 自訂建置步驟
  - 程式碼分析

所有選項(L)

```
/JMC /permissive- /GS /analyze /W4 /Zc:wchar_t /ZI /Gm- /Od /sdl /Fd"Debug\vc141.pdb" /Zc:inline
/fp:precise /D "WIN32" /D "_DEBUG" /D "_CONSOLE" /D "_UNICODE" /D "UNICODE" /errorReport:prompt
/WX /Zc:forScope /RTC1 /Gd /Oy- /MDd /FC /Fa"Debug\" /EHsc /nologo /Fo"Debug\" /Fp"Debug
\ConsoleApplication1.pch" /diagnostics:classic
```

其他選項(D) 從父代或專案預設值繼承

確定 取消 套用(A)

搜尋 英 下午 10:39 2023/1/8

# cppcheck

TEAMT5
杜 浦 數 位 安 全

- Out of bounds
- Exception safety
- Memory leaks
- Obsolete functions are used
- Invalid usage
- Uninitialized variables and unused functions
- And so on…

# cppcheck example

TEAMT5 杜浦數位安全

```
$cppcheck \
  --enable=all \
  --suppress=missingIncludeSystem \
  unused.cpp
```

yangzhixuan@yangzhixuandeMacBook-Air:~/Desktop/cpp_static_analyze/cpp... ⌥⌘1

```
➜  cppcheck cppcheck --enable=all --suppress=missingIncludeSystem unused.cpp && bat unused.cpp
Checking unused.cpp ...
unused.cpp:8:6: style: Unused variable: a [unusedVariable]
 int a;
     ^
unused.cpp:3:0: style: The function 'unused_fn' is never used. [unusedFunction]

^
```

File: unused.cpp

```cpp
#include <iostream>

int unused_fn(int a, int b) {
    return a + b;
}

int main(){
    int a;

    int b = 10;

    std::cout << b << std::endl;

    return 0;
}
```

```
➜  cppcheck
```

# cppcheck

TEAMT5 杜浦數位安全

yangzhixuan@yangzhixuandeMacBook-Air:~/Desktop/cpp_static... ⌥⌘1

```
→ cppcheck bat arr.cpp
File: arr.cpp
1  #include <array>
2  int main()
3  {
4      std::array<int, 10> arr1;
5      int arr2[10];
6      return arr1[1024];
7  }
→ cppcheck
```

yangzhixuan@yangzhixuandeMacBook-Air:~/Desktop/cpp_static... ⌥⌘1

```
→ cppcheck cppcheck --enable=all --suppress=missingIncludeSystem arr.cpp
Checking arr.cpp ...
arr.cpp:4:25: style: Variable 'arr1' can be declared as const array [constVariab
le]
    std::array<int, 10> arr1;
                        ^
arr.cpp:6:16: error: Out of bounds access in 'arr1[1024]', if 'arr1' size is 10
and '1024' is 1024 [containerOutOfBounds]
    return arr1[1024];
               ^
arr.cpp:6:12: error: Uninitialized variable: arr1 [legacyUninitvar]
    return arr1[1024];
           ^
arr.cpp:5:9: style: Unused variable: arr2 [unusedVariable]
    int arr2[10];
        ^
→ cppcheck
```

# cppcheck

TEAMT5
杜 浦 數 位 安 全

Wait, what?!

**--suppress=missingIncludeSystem**

Well …

https://stackoverflow.com/questions/6986033/cppcheck-cant-find-include-files

# clang-tidy checks UAF

TEAMT5 杜浦數位安全

yangzhixuan@yangzhixuandeMacBook-Air:~/Desktop/cpp_static... ⌥⌘1

```
→ clang-tidy bat uaf.cpp
File: uaf.cpp

#include <iostream>
#include <memory>

void foo(const int *const ptr)
{
    std::cout << *ptr << std::endl;
}

int main()
{
    auto ptr = new int(0);
    foo(ptr);
    delete ptr;

    std::cout << *ptr << std::endl;

    return 0;
}

→ clang-tidy
```

yangzhixuan@yangzhixuandeMacBook-Air:~/Desktop/cpp_static... ⌥⌘1

```
→ clang-tidy clang-tidy uaf.cpp --
1 warning generated.
/Users/yangzhixuan/Desktop/cpp_static_analyze/clang-tidy/uaf.cpp:15:18: warning:
 Use of memory after it is freed [clang-analyzer-cplusplus.NewDelete]
    std::cout << *ptr << std::endl;
                 ^~~~
/Users/yangzhixuan/Desktop/cpp_static_analyze/clang-tidy/uaf.cpp:11:16: note: Me
mory is allocated
    auto ptr = new int(0);
               ^~~~~~~~~~
/Users/yangzhixuan/Desktop/cpp_static_analyze/clang-tidy/uaf.cpp:13:5: note: Mem
ory is released
    delete ptr;
    ^~~~~~~~~~
/Users/yangzhixuan/Desktop/cpp_static_analyze/clang-tidy/uaf.cpp:15:18: note: Us
e of memory after it is freed
    std::cout << *ptr << std::endl;
                 ^~~~
→ clang-tidy
```

# However unfortunately…

TEAMT5 杜浦數位安全

yangzhixuan@yangzhixuandeMacBook-Air:~/Desktop/cpp_static... ⌥⌘1

```
→ clang-tidy bat uaf_smart.cpp

File: uaf_smart.cpp

#include <iostream>
#include <memory>

void foo(const int *const ptr)
{
    std::cout << *ptr << std::endl;
}

int main()
{
    auto ptr = std::make_unique<int>(0);

    foo(ptr.release());

    std::cout << *ptr << std::endl;

    return 0;
}

→ clang-tidy
```

yangzhixuan@yangzhixuandeMacBook-Air:~/Desktop/cpp_static... ⌥⌘1

```
→ clang-tidy clang-tidy uaf_smart.cpp --
→ clang-tidy
```

# Bad news…

```cpp
template <typename T>
class Unique_ptr
{
    using Pointer_ = T *;
    Pointer_ p = nullptr;

public:
    Unique_ptr() = default;
    explicit Unique_ptr(Pointer_ pp) : p{pp} {}
    explicit Unique_ptr(T val) : p{new T{val}} {}

    Unique_ptr(Unique_ptr &&) = default;
    Unique_ptr(const Unique_ptr &) = delete;

    Unique_ptr &operator=(const Unique_ptr &) = delete;
    Unique_ptr &operator=(Unique_ptr &&) = default;

    ~Unique_ptr() { delete p; }

    Pointer_ release() noexcept
    {
        Pointer_ pp = p;
        this->p = nullptr;
        return pp;
    }
    Pointer_ &get() noexcept { return p; }
    const Pointer_ &get() const noexcept { return p; }
};
```

https://godbolt.org/z/5ddKn4M8v

Which is rewritten in C++98 version:

https://godbolt.org/z/ebExbexWc

# CppCon 2015:

TEAMT5 杜浦數位安全

https://youtube.com/watch?v=sn1Vg8A_MPU&feature=shares&t=649

## nullptr - Null Dereferences - Conclusions

- Everyone caught the obvious one
- Only cppcheck could catch the indirect nullptr dereference
- No tools could catch the smart pointer version

# Dynamic analysis

We will introduce it in the next demo meeting.

TEAMT5
杜 浦 數 位 安 全

# Thank you.

scc@teamt5.org

TEAMT5
杜 浦 數 位 安 全