16th Hacking Camp

"C++ Secure Coding Guide"

옥찬호

Nexon Korea, Microsoft MVP

utilForever@gmail.com

소개

- 옥찬호 (Chris Ohk)
  - Nexon Korea Game Programmer
  - Microsoft VSDT MVP
    (Visual Studio and Development Technologies)
  - 페이스북 그룹 C++ Korea 대표
  - 한국청소년학술대회 KSCY 4~8회 컴퓨터 공학 멘토
  - IT 전문서 집필 및 번역 다수
    - 게임샐러드로 코드 한 줄 없이 게임 만들기 (2013)
    - 유니티 Shader와 Effect 제작 (2014)
    - 2D 게임 프로그래밍 (2014)
    - 러스트 핵심 노트 (2017)
    - 모던 C++ (2017. 9 또는 2017. 10)
    - C++17 프로그래밍 (2018?)

시작하기 전에...

- C++ 프로그래밍에서 고려해야 할 보안 문제를 살펴봅니다.
  (C 프로그래밍에서 고려해야 할 보안 문제는 분량 문제로 생략합니다.)
- C++ 언어에 대한 기본 지식이 있다고 가정합니다.
- 예상 독자 : C++로 소프트웨어 개발과 유지 보수를 하는 사람
  - 소프트웨어 취약점이 될 수 있는 프로그래밍 오류를 확인하고
    안전한 방식으로 해결할 수 있는 방법을 알 수 있습니다.
  - 나쁜 프로그래밍 습관을 들이지 않도록 해줍니다.
  - 일상적인 취약점에 대한 세부적인 내용과 발견하는 방법을 알게 됩니다.
- 모든 예제 코드는 발표 이후 Github를 통해 제공됩니다.

목차

16th Hacking Camp
C++ Secure Coding

선언 및 초기화

- C-스타일 가변 함수를 정의하지 마라.
- 레퍼런스 타입을 const나 volatile로 한정하지 마라.
- 메모리 할당 함수와 할당 해제 함수는 같은 범위에서 오버로드하라.
- 소멸자나 메모리 할당 해제 함수에서 예외를 발생시키지 마라.
- 헤더 파일에 이름 없는 네임스페이스를 정의하지 마라.
- 단정의 법칙(One Definition Rule; ODF)을 준수하라.

표현식

- 계산 순서에 따라 사이드 이펙트가 발생할 수 있는 여지를 만들지 마라.
- 계산되지 않은 피연산자의 사이드 이펙트에 의존하지 마라.
- 초기화되지 않은 메모리를 읽지마라.
- 수명이 없거나 끝난 개체에 접근하지 마라.
- 레퍼런스로 캡처된 개체가 사라지는 람다식을 정의하지 마라.
- 개체의 값 표현의 일부가 아닌 비트로 접근하지 마라.

정수

- 범위를 벗어난 열거체 값을 캐스트하지 마라.

컨테이너

- 컨테이너의 인덱스와 반복자가 유효한 범위 내에 있는지 확인하라.
- 유효한 레퍼런스, 포인터, 반복자를 사용해 컨테이너의 항목을 참조하라.
- 라이브러리 함수가 오버플로우하지 않는지 확인하라.
- 유효한 반복자 범위를 사용하라.
- 같은 컨테이너를 참조하지 않는 두 반복자에 뺄셈 연산자를 사용하지 마라.
- 결과가 오버플로우되는 경우 반복자에서 덧셈 연산자를 사용하지 마라.
- 다형성을 갖는 개체에 포인터 연산을 사용하지 마라.

문자열

- 문자열 공간이 문자 데이터와 널 문자를 담기에 충분한지 확인하라.
- 널 포인터에서 std::string을 생성하지 마라.
- 유효한 레퍼런스, 포인터, 반복자를 사용해 basic_string의 항목을 참조하라.
- 요소에 접근할 때 범위 검사를 하라.

메모리 관리

- 할당 해제된 메모리에 접근하지 마라.
- 동적으로 할당된 자원을 올바른 방법으로 할당 해제하라.
- 메모리 할당 오류를 감지하고 처리하라.
- 개체의 수명을 수동으로 관리할 경우, 명시적으로 생성 및 소멸시켜라.
- 사용자 정의 동적 할당/해제 연산자를 구현할 때, 요구 사항을 준수하라.
- 관련없는 스마트 포인터에 이미 소유한 포인터 값을 저장하지 마라.
- 지나치게 정렬된 타입의 경우, new 연산자를 사용하지 마라.

입출력

- 파일 스트림 중간에 지정된 호출 없이 교대로 입출력하지 마라.
- 파일을 더 이상 사용하지 않을 경우, 파일을 닫아라.

예외 처리

- catch 핸들러에서 가장 많이 파생된 매개 변수 타입에서 가장 적게 파생된 매개 변수 타입 순서로 정렬하라.
- 예외 규정을 준수하라.
- 예외 안전을 보장하라.
- main 함수가 실행되기 전에 발생하는 모든 예외를 처리하라.
- 실행 경계를 넘어 예외를 던지지 마라.
- 문자열에서 숫자로 변환할 때 오류를 감지하라.

개체 지향 프로그래밍

- 생성자나 소멸자에서 virtual 함수를 호출하지 마라.
- virtual 소멸자가 없는 다형성 개체를 삭제하지 마라.
- 생성자 멤버 이니셜라이저를 표준 순서로 작성하라.
- 존재하지 않는 멤버에 접근하는 .* 및 ->* 연산자를 호출하지 마라.
- C 표준 라이브러리 함수보다는 특별한 멤버 함수와 오버로딩된 연산자를 사용하라.
- 원본 개체를 변경하는 복사 동작을 만들지 마라.

동시성

- 뮤텍스에 락이 걸려있는 상태에서 파괴하지 마라.
- 예외 조건에서 걸려 있던 락이 해제되는지 확인하라.
- 여러 스레드에서 비트 필드에 접근할 때 데이터 경쟁을 방지하라.
- 미리 정의된 순서로 락을 걸어 데드락을 방지하라.
- 반복문에서 가짜로 깨어날 수 있는 함수를 감싸라.
- 조건 변수를 사용할 때 스레드의 안전성과 생존을 유지하라.

기타

- 의사 난수를 생성할 때 std::rand()를 사용하지 마라.
- 난수 발생기의 시드값이 올바른지 확인하라.
- 값을 반환하는 함수라면 모든 이탈 경로에서 값을 반환하게 하라.
- [[noreturn]]으로 선언된 함수에서는 값을 반환하지 마라.
- 시그널 핸들러는 POF(Plane Old Function)여야 한다.

감사합니다

http://github.com/utilForever

질문 환영합니다!