#include "CppUnitTest.h"

#include <string>

#include <stdexcept>

#include <memory>

#include <atomic>

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace SecurityTests

{

TEST\_CLASS(InputValidationTests)

{

public:

TEST\_METHOD(Buffer\_Overflow\_Pass)

{

char input[5];

strncpy(input, "Safe", sizeof(input) - 1);

input[4] = '\0';

Assert::AreEqual("Safe", input);

}

TEST\_METHOD(Buffer\_Overflow\_Fail)

{

char input[5];

strncpy(input, "TooLong!", sizeof(input));

input[4] = '\0';

Assert::AreNotEqual("TooLong!", input); // Expected to fail

}

};

TEST\_CLASS(SQLInjectionTests)

{

public:

TEST\_METHOD(SafeQuery\_Pass)

{

std::string userInput = "john";

std::string query = "SELECT \* FROM users WHERE name = ?";

Assert::IsTrue(query.find("?") != std::string::npos);

}

TEST\_METHOD(UnsafeQuery\_Fail)

{

std::string userInput = "' OR '1'='1";

std::string query = "SELECT \* FROM users WHERE name = '" + userInput + "'";

Assert::IsTrue(query.find(userInput) != std::string::npos); // Should be flagged

}

};

TEST\_CLASS(MemoryTests)

{

public:

TEST\_METHOD(SmartPointer\_Pass)

{

std::unique\_ptr<int> ptr(new int(5));

Assert::AreEqual(5, \*ptr);

}

TEST\_METHOD(RawPointer\_Leak\_Fail)

{

int\* ptr = new int[5];

ptr[0] = 42;

// Simulate leak (not deleting ptr)

Assert::AreEqual(42, ptr[0]);

// Memory not released — Valgrind or tools will catch this

}

};

TEST\_CLASS(ConcurrencyTests)

{

public:

TEST\_METHOD(Atomic\_Pass)

{

std::atomic<int> count(0);

count.fetch\_add(1);

Assert::AreEqual(1, count.load());

}

TEST\_METHOD(RaceCondition\_Fail)

{

int count = 0;

count++;

Assert::AreEqual(1, count); // Not thread-safe in real scenario

}

};

}