C++

Spring 2021

compscicenter.ru

Basharin Egor

t.me/egorbasharin

Lecture XIII

Security

Section 1

Variadic function

example

```
#include <cstdarg>

int max(int first, ...) {
    int res = first;
    va_list va;
    va_start(va, first);
    while (int v = va_arg(va, int)) {
        if (v > res) res = v;
    }
    va_end(va);
    return res;
}
```

printf

- %n can be used for exploits (example)

How to Avoid

- Variadic templates
- Currying

Variadic templates

```
template <class... Ts>
int my_max(int first, Ts... ts) {
    static_assert((... && std::is_same_v<Ts, int>));
    if constexpr (sizeof...(ts) == 0) {
        return first;
    } else {
        int second = my_max(ts...);
        if (first > second)
            return first;
        return second;
    }
}
```

Exceptions

- C language linkage
- Unevaluated context

C language linkage

```
// C++ code:
extern "C" void mm(int, ...);
void mm(int i, ...)
{ /* ... */ }
```

```
// C code:
void mm(int, ...);
void z()
{
    mm(10, 20);
}
```

Unevaluated context. SFINAE

```
template <class T>
struct is_f_with_strict_signature_defined {
    // might be substitution failure
    template <class Z, void (Z::*)() = &Z::f>
    struct wrapper {};

    template <class C>
    static std::true_type check(wrapper <C> * p);

    template <class C>
    static std::false_type check (...);

    static const bool value = \
        decltype(check<T>(0))::value;
};
```

Section 2

Information leakage

Example

```
#include <cstdint>
#include <string>

struct Data {
    int32_t a;
    char b;
    int32_t c;
};

void copy_to_user_buf(void* user_buffer) {
    Data data{1, 'a', 2};
    std::memcpy(user_buffer, &data, sizeof(data));
}
```

Linux Kernel Vulnerability

- CVE-2010-3881

Section 3

Uninitialized local variables

Example

```
void func(int key) {
    int s;
    switch key {
        case 1: s = 0xAA;
        case 2: s = 0xFA;
    }
    /* collapsed */
    if (s == 0xFA) {
        // ...
    }
    /* collapsed */
}
```

Vulnerabilities

- Information leakage
- Control flow

Section 4

Const object modification

Example

```
#include <optional>
int compute() { return 0; }
class A {
public:
    int value() const {
        if (!cached_) {
            const_cast<A*>(this)->cached_ = compute();
        }
        return *cached_;
    }
private:
    std::optional<int> cached_;
};
```

Risks

UB: termination, DoS-attack

Section 5

Strings & Containers

Strings

- Use std::basic_string<>
- C-strings: remember about `\0`

Bounds checking

```
void offset_it(char* dest, const char* src, size_t count, int offset) {
    for (size_t i = 0; i <= count; ++i) {
        dest[i] = src[i] + offset;
    }
}
```

Bounds checking

```
void offset_it(char* dest, const char* src, size_t count, int offset) {
    for (size_t i = 0; i <= count; ++i) { // off-by-one error
        dest[i] = src[i] + offset;
    }
}
```

Bounds checking

```
#include <vector>
#include <iostream>

std::vector<int> get_vector();

int main()
{
    std::vector<int> v = get_vector();
    for (auto b = v.begin(), e = b + 10; b != e; ++b) {
        std::cout << *b << " ";
    }
}
```

Bounds checking

```
#include <vector>

std::vector<int> get_vector();

int main()
{
    std::vector<int> v = get_vector();
    std::vector<int> dest;
    std::copy(v.begin(), v.end(), dest.begin());
}
```

Section 6

Downcasts & type confusion

Example

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

struct Base {
    virtual ~Base() = default;
};

struct A : Base {
    virtual void f(const char* s) { std::cout << "A: " << s; }
};

struct B : Base {
    virtual void g(const char* s) { std::cout << "B: " << s; }
};


int main() {
    std::unique_ptr<Base> b = std::make_unique<A>();
    static_cast<B*>(b.get())->g("Hello");
}
```

Section 7

More about memory

Example 1

```
#include <memory>

struct Data {
    int a;
    int b;
    std::string s;
};

int main() {
    Data* data;
    char* buf = new char[sizeof(data)];
}
```

Example 2

```
struct Node {
    int value;
    Node* next;
};

void free_list(Node* head) {
    for (; head != nullptr; head = head->next) {
        delete head;
    }
}
```