Foundations of Cybersecurity C/C++ Secure Coding Basics Laboratory Exercises

Michele La Manna
Dept. of Information Engineering
University of Pisa
michele.lamanna@phd.unipi.it
Version: 2022-03-24
(Original slides thanks to Eng. Pericle Perazzo, PhD)

DO STUFF_NEW

DoStuffNew Part#1

Now open the file do_stuff_new.cpp
Inspect the main, and then the part #1 of the function do_stuff_new().
What is the objective of the lazy programmer?

```
69          while(cin);
70          cin.clear();
71          cin.ignore(numeric_limits<streamsize>::max(), '\n');
72          cout << "Invoking do_stuff_new(vec)..." << endl;
73          do_stuff_new(vec);
74          cout << endl;
75      }
76      return 0;
```

```
10      int* result1 = (int*)malloc(vec.size()*sizeof(int));
11      copy(vec.begin(), vec.end(), result1);
```

10 minutes!

DoStuffNew Part#1

Now we have the vector vec, which we must consider tainted.

```
10    int* result1 = (int*)malloc(vec.size()*sizeof(int));
11    copy(vec.begin(), vec.end(), result1);
```

The lazy programmer wants to copy the content of vec in result 1.
Assume an integer wrap-around won't happen.

You have 10 minutes to find and patch the vulnerabilities.

DoStuffNew Solution#1

The lazy programmer (unbelievably so) correctly used the "copy" function.
https://www.cplusplus.com/reference/algorithm/copy/?kw=copy
However, he forgot to check if the malloc failed.

```
9       // PART #1
10      int* result1 = (int*)malloc(vec.size()*sizeof(int)); //
11      if(!result1) return;
12      copy(vec.begin(), vec.end(), result1);
```

The "malloc" function returns a null pointer if for some reasons it cannot allocate the needed amount of memory specified in the argument.

https://www.cplusplus.com/reference/cstdlib/malloc/

DO STUFF_NEW#2

DoStuffNew Part#2

What is the objective of the following code?
Figure it out!

```
13      // PART #2
14      vector<int> result2;
15      copy(vec.begin(), vec.end(), result2.begin());
16
```

5 minutes!

DoStuffNew Part#2

UNIVERSITÀ DI PISA

What is the objective of the following code?
Figure it out!

```
13      // PART #2
14      vector<int> result2;
15      copy(vec.begin(), vec.end(), result2.begin());
16
```

The lazy programmer wants to copy the content of vec in result 2.

You have 10 minutes to find and patch the vulnerabilities.

DoStuffNew Solution#2

UNIVERSITÀ DI PISA

The lazy programmer this time forgot how the "copy" function works.

```
14      // PART #2
15      vector<int> result2(vec);
16
```

The std::copy() function does not implement any bound checking, and does not expand the destination vector, so it can lead to buffer overflows.

https://www.cplusplus.com/reference/algorithm/copy/?kw=copy

DO STUFF_NEW#3

DoStuffNew Part#3

UNIVERSITÀ DI PISA

What is the objective of the following code?
Figure it out!

```
17    // PART #3
18    vector<int> result3;
19    auto tmp1 = vec.begin() + 5;
20    for(auto i = tmp1; i != vec.end(); i++)
21        result3.push_back(*i + 10); // UNCH
```

15 minutes!

DoStuffNew Part#3

The lazy programmer wants to create a copy of the array vec from the 6th element to the last. The duplicated element should have its value increased by ten.

```
17      // PART #3
18      vector<int> result3;
19      auto tmp1 = vec.begin() + 5;
20      for(auto i = tmp1; i != vec.end(); i++)
21          result3.push_back(*i + 10); // UNCH
```

Example vec= [1 2 3 4 5 6 7 8 9 10]
Result3 = [16 17 18 19 20]

You have 5 minutes to find and patch the vulnerabilities.

DoStuffNew Solution#3

UNIVERSITÀ DI PISA

What happens if vec is shorter than 5?
Then vec.begin()+5 has no meaning! It will be forever different from vec.end!

```
17      // PART #3
18      vector<int> result3;
19      auto tmp1 = (vec.size() < 5) ? vec.end() : (vec.begin() + 5);
20      for(auto i = tmp1; i != vec.end(); i++) // UNCHANGEABLE!
21          result3.push_back(*i + 10); // UNCHANGEABLE!
22
```

DO STUFF_NEW#4

DoStuffNew Part#4

UNIVERSITÀ DI PISA

What is the objective of the following code?
Figure it out!

```
23      // PART #4
24      vector<int> result4(vec); // UNCHANGEABLE!
25      auto tmp2 = result4.end() - 2;
26      for(auto i = result4.begin(); i != tmp2; i++){
27          if(*i % 2 == 0){
28              result4.insert(i, *i + 20);
29              i++;
30          }
31      }
```

15 minutes!

DoStuffNew Part#4

UNIVERSITÀ DI PISA

The lazy programmer wants to duplicate the even-valued elements of the vector from the beginning to the second to last element. The duplicated element has its value increased by 20.

```
23      // PART #4
24      vector<int> result4(vec); // UNCHANGEABLE!
25      auto tmp2 = result4.end() - 2;
26      for(auto i = result4.begin(); i != tmp2; i++){
27          if(*i % 2 == 0){
28              result4.insert(i, *i + 20);
29              i++;
30          }
31      }
```

result4= [1 2 3 4 5 6] → result4 = [1 2 **22** 3 4 **24** 5 6]

You have 5 minutes to find and patch the vulnerabilities.

DoStuffNew Solution#4

UNIVERSITÀ DI PISA

What happens if vec is shorter than 2?
Then vec.end()-2 has no meaning! It will access out of bound!

```
23      // PART #4
24      vector<int> result4(vec); // UNCHANGEABLE!
25      size_t tmp_size = result4.size();
26      auto tmp2 = (tmp_size < 2) ? result4.end() : (result4.end() - 2);
27      for(auto i = result4.begin(); i != tmp2; i++){ // UNCHANGEABLE!
28          if(*i % 2 == 0){
29              i = result4.insert(i, *i + 20);
30              i++;
31              tmp2 = (tmp_size < 2) ? result4.end() : (result4.end() - 2);
32          }
33      }
34
```

Furthermore, the lazy colleague forgot that the "insert()" function can change the size of the vector, which in turn invalidates all the existing iterators of that vector!
You must re-assign all the iterators that refers to result4, **including** "tmp2"!

DO STUFF_NEW_FINAL

do_stuff_new_final

UNIVERSITÀ DI PISA

```
30  int main() {
31      string str1;
32      string str2;
33
34      cout << "Welcome to do_stuff_new_final() invoker." << endl;
35      cout << endl;
36
37      for(;;){
38          cout << "str1 = ";
39          getline(cin, str1);
40          if(!cin) return 0;
41          cout << "str2 = ";
42          getline(cin, str2);
43          if(!cin) return 0;
44          //cin.ignore(numeric_limits<streamsize>::max(), '\n'); // clear cin buffer
45          cout << "Invoking do_stuff_new_final(" << str1 << "," << str2 << ")..." << endl;
46          do_stuff_new_final(str1, str2);
47          cout << endl;
48      }
49      return 0;
50  }
51
```

PART #1

do_stuff_new_final Part#1

UNIVERSITÀ DI PISA

Remember, the parameters "str1", "str2" must be considered as **tainted**.

```
7  void do_stuff_new_final(const string& str1, const string& str2){
8  // NOTE: For esoteric reasons, some code lines are UNCHANGEABLE.
9      // PART #1
10     if(str1.empty()) return;
11     string command = "ping -c 1 " + str1; // UNCHANGEABLE!
12     int result1 = system(command.c_str()); // UNCHANGEABLE!
```

What is the objective of this function?

5 minutes!

do_stuff_new_final Part#1

UNIVERSITÀ DI PISA

What is the objective of the following code?
Figure it out!

```
7  void do_stuff_new_final(const string& str1, const string& str2){
8  // NOTE: For esoteric reasons, some code lines are UNCHANGEABLE.
9      // PART #1
10     if(str1.empty()) return;
11     string command = "ping -c 1 " + str1; // UNCHANGEABLE!
12     int result1 = system(command.c_str()); // UNCHANGEABLE!
```

The lazy programmer wants to send a ping to the address specified in "str1". **Is it safe? Why not?**
"*Because it is an exercise on secure coding*", although correct, is not a valid answer.

You have 10 minutes to find and patch the vulnerabilities.

do_stuff_new_final Solution#1

UNIVERSITÀ DI PISA

To avoid dangerous **OS command injections**, we have to sanitize str1.
How can we sanitize it? The recommended approach is whitelisting.

```
8  void do_stuff_new_final(const string& str1, const string& str2){
9   // NOTE: For esoteric reasons, some code lines are UNCHANGEABLE.
10      // PART #1
11      if(str1.empty()) return;
12      static char ok_chars[] = "abcdefghijklmnopqrstuvwxyz"
13                               "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
14                               "1234567890-.:"; // '-' is allowed i
15      if(str1.find_first_not_of(ok_chars) != string::npos) return;
16      if(str1[0] == '-') return; // avoid option injection by black
17      string command = "ping -c 1 " + str1; // UNCHANGEABLE!
18      int result1 = system(command.c_str()); // UNCHANGEABLE!
19
```

str1 specifies an IPv4 address, an IPv6 address, or an hostname.
The character "-" is allowed in an hostname, but it could also be used to inject some unwanted option in the ping command.
This is avoided by line 16 (although it follows the blacklist approach).

PART #2

do_stuff_new_final Part#2

UNIVERSITÀ DI PISA

What is the objective of the following code?
Figure it out!

```
14    // PART #2
15    ifstream f(str2, ios::in); // only files in the home directory or its subdirs should be opened here!
16    if(!f) { cerr << "Cannot open " << str2 << endl; return; }
17    cout << "Content of " << str2 << ":" << endl;
18    string line;
19    do{
20        getline(f, line);
21        cout << line << endl;
22    }
23    while(!f.eof());
24    f.close();
```

5 minutes!

do_stuff_new_final Part#2

UNIVERSITÀ DI PISA

What is the objective of the following code?
Figure it out!

```
14    // PART #2
15    ifstream f(str2, ios::in); // only files in the home directory or its subdirs should be opened here!
16    if(!f) { cerr << "Cannot open " << str2 << endl; return; }
17    cout << "Content of " << str2 << ":" << endl;
18    string line;
19    do{
20        getline(f, line);
21        cout << line << endl;
22    }
23    while(!f.eof());
24    f.close();
```

The lazy programmer wants to redirect to the output stream the content of a file that is in "/home/" or in one of its sub-directories.

You have 10 minutes to find and patch the vulnerabilities.

CANONICIZ…
CANONICITA…
CANONICALITA…

do_stuff_new_final Solution#2

UNIVERSITÀ DI PISA

Canonicalization.
This control is made in two phases:
1) invoke the "realpath()" function on str2. Remember that realpath wants a c-string, so use the ".c_str()" class member function on str2.
2) Check that the first segment of the path is the directory "home".

```
21  char* canon_str2 = realpath(str2.c_str(), NULL);
22  if(!canon_str2) return;
23  if(strncmp(canon_str2, "/home/", strlen("/home/")) != 0) { free(canon_str2); return; }
24  ifstream f(canon_str2, ios::in); // only files in the home directory or its subdirs sho
25  free(canon_str2);
26  if(!f) { cerr << "Cannot open " << str2 << endl; return; }
27  string line;
```

WARNINGS:

- The symbol "~" is interpreted as "/home" by the shell, and not by the compiler. If you replace "/home/" with "/~/" in line 23, the code won't work properly.

- A very gifted adversary is *theoretically* able to create a symbolic path in the window of time between lines 21 and 24. (TOC TOU problem)

Laboratory Material

UNIVERSITÀ DI PISA

Shortly after each laboratory, the slides and the solutions will be uploaded to the "Files" Tab of our MS Teams "General" channel.

See you whenever!