Feedback — Week 2

Help

You submitted this quiz on **Sun 9 Nov 2014 10:44 PM PST**. You got a score of **45.00** out of **68.00**. You can attempt again, if you'd like.

Question 1

When could an integer overflow impact memory safety?

Your Answer		Score	Explanation
If the integer was passed as a parameter to open()	~	1.00	open does not use its integer parameters to access memory
Integer overflows always impact memory safety	~	1.00	Integer overflows can be by design in some algorithms and only impact memory safety when the integer is used in a way that interacts with memory
If the integer was passed as a parameter to <pre>printf()</pre>	~	1.00	printf does not use its integer parameters to access memory
If the integer was used to index into an array	~	1.00	then the integer value may not be correct when indexing into memory, e.g., if it was unsigned, and the overflow caused it to be negative

If the integer is passed as an argument to malloc()	×	0.00	then the integer value passed to malloc could differ from the integer used to iterate over the buffer (e.g., it could have been multiplied by a data size)
Total		4.00 /	
		5.00	

Question 2 A program indexes a buffer after a pointer to that buffer has been used as a parameter to the free() function. This is **Your Answer Explanation** Score Correct behavior An information flow violation A violation of spatial memory safety A violation of temporal memory safety 🗸 4.00 Use of a buffer beyond its lifetime is a temporal safety issue Total 4.00 / 4.00

A language that uses garbage collection for memory management:

Your Answer		Score	Explanation
Will not allow temporal memory safety violations	~	3.00	The garbage collector will ensure that memory is only deallocated when it is not reachable, and this decision is not left up to the programmer
Will not allow type safety violations			
Will not allow spatial memory safety violations			
All of these			
None of these			
Total		3.00 / 3.00	

Question 4

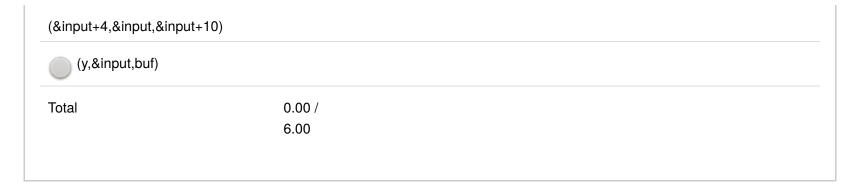
Consider the following code:

```
char *foo(char *buf) {
   char *x = buf+strlen(buf);
   char *y = buf;
   while (y != x) {
      if (*y == 'a')
         break;
      y++;
   }
   return y;
}

void bar() {
   char input[10] = "leonard";
   foo(input);
}
```

The definition of spatial safety models pointers as capabilities, which are triples (p,b,e) where p is the pointer, b is the base of the memory region the pointer is allowed to access, and e is the extent of that region. Assuming characters are 1 byte in size, what is a triple (p,b,e) for the variable y when it is returned at the end of the code?

Your Answer		Score	Explanation
(&input+4,&input,&input+7)	×	0.00	While the length of the string is 7 characters, the full extent of the buffer is 10 characters, per the declaration of input[]
(&input+4,0,sizeof(input))			



Select all that apply. A type-safe language:

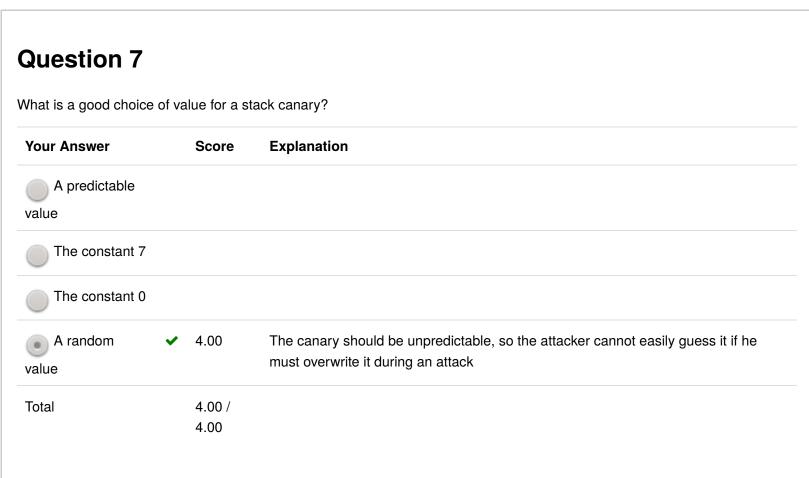
Your Answer		Score	Explanation
Is always <i>much</i> slower than a non-type safe language	~	1.00	Some type-safe languages are much slower, but not all. Type-safe languages can be optimized to run within a couple of factors of C and/or C++, and even better when applied to program domains for which they were designed
Is sometimes memory safe, but not always	×	0.00	Type safe languages are always memory safe
Is also memory safe	×	0.00	Type safety is stronger than memory safety
Can be used to enforce information flow	~	1.00	This is done in the JIF programming language

security			
Total	2.00 / 4.00		
	4.00		

An engineer proposes that in addition to making the stack non-executable, your system should also make the heap non-executable. Doing so would

Your Answer	Score	Explanation
Not make the program more secure,		
because attacker-controlled data cannot be		
stored in the heap		
Ensure that only the correct amount of		
data was written to a heap-allocated block,		
preventing heap overflows		
Make the program more secure by	✓ 4.00	Then attacker data in the heap cannot be executed,
disallowing another location for an attacker to		enforcing (W xor X) / DEP for the entire program
place executable code		





A return-to-libc attack does not require that the attacker inject executable code into the vulnerable program. Which of the following is the *most important* reason that return-to-libc attacks are useful to the attacker?

Your Answer	Score	Explanation
The code in libc is better than code the attacker would write		
There is no need to modify the application's executable code	0.00	The attacker can compromise the program without modifying the applications executable code; code injection attacks, for example, do not modify the existing code
The injected code might have bugs		
There is no need to be able to execute (writable) data		
Total	0.00 / 5.00	

a return-oriented program (ROP), what is the	he role	of the sta	ck pointer?
our Answer		Score	Explanation
It's like the program counter in a normal program	~	4.00	the stack pointer is used to select the next instruction to execute via a 'ret'
It's like the allocation pointer used by malloc()			
It's really no different than in a normal program			
It's like the frame pointer in a normal program			
Total		4.00 / 4.00	

When enforcing Control Flow Integrity (CFI), there is no need to check that direct calls adhere to the control flow graph

our Answer		Score	Explanation
CFI should be deployed on systems that			
ensure the data is non-executable			
Programs that use CFI don't have direct	×	0.00	Most programs have direct calls, and CFI ought to (and
	•••	0.00	does) apply to most (or all) programs
ealls			,,,,
CFI should be deployed on systems that			
ensure the code is immutable			
The attacker is not interested in corrupting			
lirect calls			
- Total		0.00 /	
		4.00	

Recall that classic enforcement of CFI requires adding labels prior to branch targets, and adding code prior to the branch that checks the label to see if it's the one that is expected. Now consider the following program:

int cmp1(char *a, char *b) {

```
return strcmp(a,b);
int cmp2(char *a, char *b) {
  return strcmp(b,a);
}
typedef int (*cmpp)(char*,char*);
int bar(char *buf) {
  cmpp p;
  char tmpbuff[512] = { 0 };
  int 1;
  if(buf[0] == 'a') {
    p = cmp1;
  } else {
    p = cmp2;
  printf("%p\n", p);
  strcpy(tmpbuff, buf);
  for(1 = 0; 1 < sizeof(tmpbuff); 1++) {</pre>
    if(tmpbuff[1] == 0) {
      break;
    } else {
      if(tmpbuff[1] > 97) {
        tmpbuff[1] -= 32;
```

```
}
}
return p(tmpbuff,buf);
}
```

To ensure that the instrumented program runs correctly when not being attacked, which of the following functions would have to be given the same label? Choose at least two, but no more functions than necessary.

Your Answer		Score	Explanation
printf	~	1.00	cannot be assigned to p, a function pointer and therefore an indirect branch target
cmp1	~	2.00	could be assigned to p, a function pointer and therefore an indirect branch target
strcpy	~	1.00	cannot be assigned to p, a function pointer and therefore an indirect branch target
bar	×	0.00	cannot be assigned to p, a function pointer and therefore an indirect branch target
cmp2	~	2.00	could be assigned to p, a function pointer and therefore an indirect branch target
Total		6.00 / 7.00	

Question 12

In your review of a program, you discover the following function:

```
void aFunction(char *buf) {
    static char BANNED_CHARACTERS[] = {'>', '<', '!', '*'};
    int 1 = strlen(buf);
    int i;

for(i = 0; i < 1; i++) {
      int j;
      int k = sizeof(BANNED_CHARACTERS) / sizeof(char);
      for(j = 0; j < k; j++) {
         if(buf[i] == BANNED_CHARACTERS[j])
         buf[i] = ' ';
      }
    }
}</pre>
```

How would you best describe what this function is doing?

ation
entially dangerous ("black") character, given in the list, is present s removed

Spatial safety

enforcement

Input validation by whitelisting	ру			
Total	6.00	/		
	6.00			

safe string library typically attempts to ensure which of the follow	wing?		
Your Answer		Score	Explanation
That the strings have been properly sanitized			
That strings from the safe library can be freely passed to			
he standard string library functions, and vice versa			
That there is sufficient space in a source and/or target	~	4.00	safe string libraries enforce spatial
string to perform operations like concatenation, copying, etc.			memory safety
That wide (i.e., multibyte) character strings can be used			
where single-byte character strings are expected.			
Total		4.00 /	
		4.00	

A project manager proposes a C coding standard where pointer variables must be assigned to NULL after being passed to

our Answer	Score	Explanation
Helps code readability, but not security		
Stops writes to stale pointer values that might otherwise succeed		
nd result in program compromise		
Is a poor security decision, because NULL pointer dereferences	× 0.00	Crashes are better than
ould cause the program to crash		compromise
Prevents memory leaks, thus avoiding potential denial of service		
ıtal	0.00 /	
	4.00	

A colleague proposes using a heap allocator that randomizes the addresses of allocated objects. This:

Your Answer		Score	Explanation
Will make the program more secure, because attackers frequently rely on predicting the locations of heap-allocated objects in exploits	~	4.00	

Will increase performance by keeping the cache sparsely performance by k	opulated
Will make the program less secure, because the application predict the locations of heap-allocated objects	will not be able to
Will have no impact on security or performance	
Total	4.00 / 4.00

https://class.coursera.org/softwaresec-001/quiz/feedback?subm...