CS-412 Final Exam (Spring 2021)

SCIPER:_____

Last name:_____

First name:_____

1. This is a take home open book exam. While you are allowed to use any available material, communication with other people is strictly forbidden.
2. You have 120 minutes (from 10:15 to 12:15), and there are 100 points. Use the number of points as guidance on how much time to spend on each question.
3. Only in the case of an Internet outage or other emergency, you are allowed to take pictures of your solution sheet and submit your exam by sending those pictures to the instructor (mathias.payer@epfl.ch).
4. If you have a question, join the provided Zoom meeting and wait for the proctor. Do not communicate to other students.Any announcements and corrections will be provided on Moodle.
5. State all extra assumptions that you make in addition to those stated as part of a question.
6. The course staff is aware of the potential for abuse and cheating during this unusual take-home examination and will carefully analyze submissions to detect similarities in responses in addition to other types of correlations. We may require some students to defend and justify their take home exam solutions in an oral exam. The outcome of the oral exam may then replace the score from the take home exam.
7. By signing the following form, you certify that you solved these problems on your own, that you turn in your solution, and that there were no environmental or other factors that disturbed you during this exam or that diminished your performance.

Signature:_____

Q1) First-Person Compartmentalizer

Consider the following scenario:
An online, multiplayer game requires interaction with game assets on a disk, network access to send and receive player data, GPU access to render the video, and a core module for game logic. The game has a server-client architecture, where the remote server stores the global game state, and the local client runs the game from a particular user's viewpoint. The user is the person playing the game on their local desktop.

What is compartmentalization in general (describe in your own words, 1-2 sentences)? [1]

Why is compartmentalization useful for improving the security of this program? [1]

What are the inputs and outputs for the client? [1]

When writing the client program, what is the attacker model that the programmer should defend against? [1]

For this attacker model, which interfaces are untrusted, and why? [1]

How would an attacker try to exploit the client using this untrusted interface? [1]

What is a reasonable defense? [1]

How do you leverage compartmentalization to reduce the attack surface, given the attacker model from the previous question? [2]

Describe a meaningful attacker model for the server program. [1]

For this attacker model, which interfaces are untrusted, and why? [1 + 1]

How would an attacker try to exploit the server using this untrusted interface? [1]

Can compartmentalizing the server protect it from hackers who misrepresent their actions in the game (eg. aimbot in a shooting game)? Justify. [1]

Q2) The queen’s CIA

You have been tasked by your Queen to manage the pigeon messaging system between the Queen in her main castle and
her army.
For each of these statements, select and justify which of the following property is being applied:
Confidentiality - Integrity - Availability - None of the above

1. You only choose the most reliable pigeons with the best sense of orientation. [2]

2. Every message sent is translated in a language only known to the statespeople of your queendom. [2]

3. Every message is sent in an envelope sealed with a wax seal made with the Queen's emblem. [2]

While these measures increase the security of the communications, attack vectors still exist. For each of the measures listed above, provide one attack vector that could defeat the property. [3]

Q3) Mitigation reversing

Please consider the following assembly snippet (in AT&T syntax) for the questions listed on this page.

```
mov %rsp,%rbp
sub $0x10,%rsp
mov $0xdeadbeef,%rax
nop
nop
mov %rax,-0x8(%rbp)
lea 0xe98(%rip),%rdi # points to string "Hello World!"
xor %ecx,%ecx
mov %cl,%al
call 1040 <printf@plt>
mov $0xdeadbeef,%rdx
nop
nop
mov -0x8(%rbp),%rsi
cmp %rsi,%rdx
jne 1191 <fun+0x41>
add $0x10,%rsp
pop %rbp
ret
call 1030 <__stack_chk_fail@plt>
cs nopw 0x0(%rax,%rax,1)
```

Which kind of common mitigation is used/active in the above snippet? [2]

This implementation has a particular flaw. Describe the flaw and how you would exploit it. [2]

Some implementations of this mitigation include a NULL byte. What particular attack angle do NULL bytes prevent? [1]

Assume that you are protecting a web server that spawns a thread for each new request. How do you need to adjust the defense above so that the mitigation protects each request (in the spirit of the mitigation)? When is the extra code executed (e.g., at the start of the process, when new threads are created, when a request is handled)? [1 + 1]

Why does CFI on the backward edge alone not make sense? [2]

What advantages does C++ class hierarchy analysis CFI have in comparison to function prototype CFI? Discuss the potential security benefits if any. [2]

Q4) The CFI details

Please refer to these function definitions for the question listed on the rest of this page.

```
float add3(float x, float y, float z);
float norm3(float x, float y, float z);
int rand(void);
void srand(int seed);
int inv_mod(int val, int mod);
float div(float x, float y);
float greater(int x, int y);
char* hello_world(void);
int getchar(void);
bool free(void*);
void* memcpy(void* dest, void* src, int n);
char* strncat(char* dest, char* src, int n);
int puts(char* str);
void print(char* str);
double tan(double arg);
double sqrt(double arg);
void print_upto(char* str, int idx);
int split_in_place(char* str, int idx, int x);
int strcmp(char* str1, char* str2, bool flag);
char* strcat(char* dest, char* src);
char* vuln(char* arg1, char* arg2, int* arg3);
```

The code contains an indirect call using the following function pointer.

`char * (*foo)(char *, char *, int);`

For each of the three following CFI techniques (single set, argument arity, function prototype), list the valid function targets when calling using foo.

Single set CFI [1]

Argument arity CFI [3]

Function prototype CFI [3]

An attacker wants to redirect the call using foo to execute the vulnerable function vuln.

```
char* vuln(char* arg1, char* arg2, int* arg3);
```

Which CFI technique would successfully thwart the attacker? [1]

Q5) CTF by hand

Note that this question is hard and likely requires some thought before you can get the points. Attempt this question at the end. The following listing shows the assembly code for function foo (in AT&T syntax). Understand this code, and answer the questions below.

```
xor %rdx,%rdx
mov $0x1,%r8
cmp %rsi,%rdx
je 41 <foo+0x2a>
xor %rcx,%rcx
xor %rax,%rax
cmp %rdi,%rcx
je 39 <foo+0x22>
add %r8,%rax
inc %rcx
jmp 2c <foo+0x15>
inc %rdx
mov %rax,%r8
jmp 23 <foo+0xc>
mov %r8,%rax
retq
```

This function foo is part of a CTF program shown below. This program calls foo and uses the return value to generate and print the flag.

```
int poss[] = {
31, 30, 28,
27, 26, 25,
23, 21,
19, 18, 16,
15, 13, 12,
11, 10, 9,
7, 6, 5,
3, 2, 1, 0
};
int main() {
     unsigned long flag = 0;
     for (unsigned i = 0; i < sizeof(poss)/sizeof(poss[0]);i++)
          flag += foo(2, poss[i] + 32);
          printf("%lx\n", flag);
}
```

Write the function declaration for foo. [1]

Are the arguments/return values signed or unsigned? How can you tell? [1]

What does it compute, and explain how it does so? [4]

Write the C code which would correspond to this function. [4]

Comment the code to point to the assembly address for each block. [1]

1. What is the flag? [1]

2. The program has one particular issue which will prevent you from getting the flag by just running it. What is this issue? [2]

3. How will you fix the program to get the flag in a CTF? [1]

4. Write an alternate version of foo (in assembly/C) which will allow you to get the flag. [2]

If foo were used in a real program, it would be vulnerable due to a particular type of bug. [1]

What is this bug, and which sanitizer could detect it? [1]

Q6) The matter of orienting your programming

Describe the gadgets used in a ROP chain. [1]

What do all ROP gadgets require? [1]

Describe the gadgets used in a JOP attack. [1]

What is the key gadget required to execute a JOP attack called? [1]

Find ROP gadgets from this code snippet (Intel syntax). You can use https://defuse.ca/online-x86-assembler.htm for easy disassembly or use your own. [3]

```
0: b8 c3 d1 00 00     mov eax,0xd1c3
5: ff c3              inc ebx
7: e8 03 f3 c3 00     call 0xc3f308
```

Q7) Sanitizers and testing

How are sanitizers useful during testing? What policy does ASan enforce? Describe two of its limitations. [1+1+2]

Describe a limitation of dynamic testing techniques in general and explain (giving an example) how it applies to fuzzing. [1+2]

What is state explosion for symbolic execution and how can it be constrained? [1+2]

For each of the following, give a set of inputs that gives

1. full statement coverage [1],
2. full branch coverage [2],
3. full path coverage [1].

If you can't define full coverage, state why this is not achievable.

```
void func(int a, int b, int c) {
      int x = 12, y = 5, z = 0;
      while (a != 0) {
            if (b == 12) break;
            switch (c) {
                  case 0: x++;
                  case 1: z++;
                  default: a--;
            }
      }
}
```

Fuzzers like the American Fuzzy Lop (AFL) rely on a global coverage map to determine if new seeds are interesting. Why can it be said that the longer AFL runs, the more it approaches blackbox fuzzer behavior? [2]

Fuzzing file-based targets requires restarting them for every generated test case. This incurs a high startup overhead and reduces fuzzing throughput. To mitigate this, AFL employs a set of optimizations. List and describe three of them. [5]

1.

2.

3.

AFL uses a hashmap to store coverage information. List two advantages and the key disadvantage of using a hashmap. [1+1+1]

Q8) Android Security Team

How does Android leverage the Linux kernel's user-based permissions to enforce isolation? [3]

How does Android protect against tampering of application files? What is the threat model in this situation? [1+1]

Why are low level memory safety vulnerabilities still a problem despite apps running on top of a memory-safe VM? [2]

How is access to sensitive services (e.g., camera, location, or telephony) by applications mediated by the Android OS? [1]