UNIVERSITY OF BRISTOL

MOCK Examination Period

FACULTY OF ENGINEERING

Examination for the Degrees of BSc, BEng, MEng and MSc

COMSM0050
Systems & Software Security

TIME ALLOWED:
2 Hours

Calculators must have the Faculty of Engineering Seal of Approval.

TURN OVER ONLY WHEN TOLD TO START WRITING

Short Questions (5 marks each)

1. Explain briefly what the dirty COW vulnerability is. (5pt)
2. Explain what the YAMA LSM is. (5pt)
3. Explain briefly how Linux and Windows access control differ. (5pt)
4. Discuss the pros and cons of anomaly-based versus signature-based intrusion detection. (5pt)
5. In the context of malloc *size* (as its parameter), how will you check if the size calculation may contain integer overflow? (5pt)
6. In the article by Shacham *"The Geometry of Innocent Flesh on the Bone: Return-into-libc without Function Calls"*, in the *section 1.2.5* ***Wait, What about Zero Byte***, what problem do you see when the address of a particular gadget contains \00? (5pt)
7. Answer the following questions on Fuzzing:
    a. Why fuzzing is more challenging for networked/server type applications? (2pt)
    b. What was the main advantage of using taint flow analysis on fuzzing as described in VUzzer paper? (3pt)
8. Consider the code given below. There is a bug that allows standard stack smashing type exploit. Explain the bug and how is can result in stack smashing? (5pt)

```
int main(int argc, char *argv[]){
        unsigned short s;
        int i;
        char buf[800];
        if(argc < 3){
            return -1;
        }
        i = atoi(argv[1]);
        s = i;
        if(s >= 800){
            printf("Too long argument!\n");
            return -1;
        }
        printf("s = %d\n", s);
        memcpy(buf, argv[2], i);
        printf("%s\n", buf);
        return 0;
 }
```

Long Questions (15 marks each)

1. Describe in detail how you would implement a kernel rootkit. (15pt)
2. Discuss potential mechanism(s) to detect kernel rootkits or to prevent them from taking hold in a system. (15pt)
3. Answer the following question on defence:
   a. Why does C++ code impose more challenges for a *fine-grained* CFI when compared to C code? (4pt)
   b. Why does backward CFI is challenging to implement? (7pt)
   c. What is ASLR and how does it help in mitigating certain attacks? (4pt)
4. Answer the following questions on program Analyses:
   a. What are shortcomings of static analysis that a dynamic analysis can address? (5pt)
   b. How can you implement static dataflow analysis to detect uninitialized access to variables/memory? (10pt)

This is the end of the exam.