Computer Security

Prof. Dr.-Ing. Volker Roth
Freie Universität Berlin

July 4, 2012

Please note that your submission will not be accepted without the properly signed Academic Integrity agreement.

Academic Integrity

We each certify that this submission is our own original work and that we have due acknowledged any work of others.

_____

Date, signature, name in block letters

_____

Date, signature, name in block letters

_____

Date, signature, name in block letters

Question 1: TOCTOU

The following C snippet is a sample TOCTOU vulnerability from an old Linux version of `lpr`:

```
1 for (int i=1; i < argc; i++) {
2   if (!access(argv[i], O_RDONLY)) {
3     fd = open(argv[i], O_RDONLY);
4   }
5   print(fd);
6 }
```

Assume that for some reason `lpr` runs with suid root.

1. Show how this program can be abused to print the /etc/shadow password file which is usually only readable by root. Use filesystem operations (mkdir, touch, unlink, etc.) for illustration and state precisely where your commands have do be executed to exploit the program successfully.
2. Explain concisely and precisely in one sentence what *hardness amplification* means and how it is be applied to making the exploitation of file system races harder.

Question 2: Signedness Bug

The following code contains a *Signedness Bug*. Spot the problem, state a problematic input and explain in detail what happens.

```
1  #include <stdio.h>
2  #include <string.h>
3
4  void beHappy(int happyFactor) {
5     int maxHappyNess = 64;
6     char outPut[maxHappyNess+8+1];
7     char happy[] = "im happy";
8     strncpy(outPut, happy, 8);
9
10    if(happyFactor < maxHappyNess) {
11       unsigned int i;
12       for(i=0; i < happyFactor; i++) {
13          outPut[7+i] = 33;
14       }
15       outPut[7+happyFactor] = 0;
16    }
17    else {
18       outPut[8] = 0;
19    }
20
21    printf("%s\n", outPut);
22 }
23
24 int main(int argc, char *argv[]) {
25    if(argc != 2) {
26       return -1;
27    }
28
29    beHappy(atoi(argv[1]));
30 }
```

Question 3: Format String Vulnerabilities

Assuming that you are on a 32bit machine, given the code snippet below answer the following questions.

1. How can you find the relative Position of the return address? Sketch a Format String.
2. Give a Format String that reads from address 0x08048567
3. Given an uncertainty of 32 Bytes about the position of the program in memory
   (a) What's the maximum size of a shell code you could use if your looking for an absolutely reliable exploit? Sketch the Format String.
   (b) Draw an image of the stack directly before and during execution of your exploit. Mark all relevant addresses and elements.

```
1  void superFunction(char* user){
2        char buffer[256];
3        char tmpBuffer[512];
4        char inBuffer[128];
5
6        toUpperCase(tmpBuffer);
7
8        sprintf (buffer, "Result is: %100s", user);
9        sprintf (inBuffer, buffer);
10 }
```