# Computer and Network Security
10th of June 2014

---

- This exam consists of 5 questions with subquestions. Unless indicated otherwise, every subquestion counts for 10 points.
- Mark every page with name and student number.
- Use of books, calculator, or additional course material is prohibited.
- Always explain your answers. At the same time, keep your answers short and to the point. Do not use pencil or red ink.
- Do not panic if you think the exam is (too) difficult - this year it will be graded on a curve.

---

1. **True/False with Reason (10 points total).** For each of the statements below, state whether they are true or false. Explain your answer with a short justification, at most 2 sentences long.
   1. Assuming the location of the executable is not randomized, can any shellcode-based attack be made into a ROP-based attack to bypass no-execute restrictions (regardless of the program that is attacked)?
   2. A vulnerable application filters out the 0xcd byte that represents the INT instruction used for trapping into the kernel. There are no other opcodes available that trap into the kernel and you don't know (and cannot brute-force) the location of the entry points into libc. Can you still use shellcode to create a shell?
   3. A security expert investigates the university's (packet-based) IDS which has 99.9% accuracy. In other words, it looks at every network packet and if it is malicious, the IDS will flag it as such in 99.9% of the cases. Conversely, it will erroneously flag only 0.01% of all benign traffic as malicious. Say the VU Campus receives about 25 million packets per day and on average one packet per day is malicious. The expert thinks for a moment and says: "So... if the IDS raises an alarm, the probability of it being a malicious packet is about 0.4%". True or False? (No calculator needed.)
   4. Comparing botnet command-and-control (C&C) structures, one could say that IRC-based botnets are relatively easy to detect for network-based intrusion detection systems, while HTTP-based botnets are much harder to spot, and p2p-based botnets are even harder still.
   5. Electronic code books are great to prevent block insertion and block deletion in block ciphers.
2. **Memory errors** A SUID root program has the following code:

```c
#include<stdio.h>
#include<string.h>

int main(int argc, char **argv) {
   char *p, result[512];
   char hostname[128], password[128], username[128];

   if (argc != 4) {
     fprintf(stderr, "bad arguments\n");
     return -1;
   }

   strcpy(username, argv[1]);
   strcpy(password, argv[2]);
   strcpy(hostname, argv[3]);

   p = result;
   strcpy(p, "ftp://"); p += 6;
   strcpy(p, username); p += strlen(username);
20      strcpy(p, ":");      p += 1;
21      strcpy(p, password); p += strlen(password);
22      strcpy(p, "@");      p += 1;
23      strcpy(p, hostname);
24
25      printf("%s\n", result);
26      return 0;
27    }
```

(a) Explain why this code is vulnerable and how you can exploit it to launch a shell. Assume that all possible protections (such as address space layout randomisation, bound checking, canaries and non-executable stack) are disabled and that variables are pushed onto the stack in the order in which they are declared with no additional padding. You should describe your inputs in enough detail to make it work but there is no need to give the exploit code or shellcode.

(b) After finding his entire user database posted on Pastebin by the famous CLF group, the system administrator finds the vulnerability and attempts to fix the program by replacing the argument check (lines 8–11) with a more elaborate validation:

```
    if (argc != 4 ||
      strlen(argv[1]) > sizeof(username) ||
      strlen(argv[2]) > sizeof(password) ||
      strlen(argv[3]) > sizeof(hostname)) {
      fprintf(stderr, "bad arguments\n");
      return -1;
    }
```

How would you exploit the program now?

(c) Assuming the stack is made non-executable, is it still possible to exploit the program? If so, what do you need to change?

(d) Assuming stack canaries are also used, is it still possible to exploit the program? If so, what do you need to change?

3. **Control Flow Integrity, SROP and ROP**

(a) Control flow integrity (CFI) ensures that indirect branches can only reach valid targets. Will CFI be able to stop the control flow diversion caused by sigreturn oriented programming? Explain!

(b) On the surface, ROP and SROP are similar, but they have different pre-conditions (and also different effects). Explain the differences between the pre-conditions for both techniques in the case of a 64 bit exploit (as discussed in the paper). Also explain which pre-conditions you think are easier to satisfy and why.

4. **Networks and such**

The police raids your pirates' nest. On the terminal of one of your computers the officers read:

```
Source        Destination   Summary
--------------------------------------------------------------------------------------
[14.10.0.10] [31.3.3.7] FTP: C PORT=37205   PORT 192,168,0,5,0,93
[31.3.3.7] [14.10.0.10] FTP: R PORT=37205   200 PORT command successful.
[14.10.0.10] [31.3.3.7] FTP: C PORT=37205   LIST
[31.3.3.7] [192.168.0.5] TCP: D=XX S=XX SYN SEQ=474501024 LEN=0 WIN=65535
[192.168.0.5] [31.3.3.7] TCP: D=XX S=XX RST ACK=474501025 WIN=0
[31.3.3.7] [14.10.0.10] FTP: R PORT=37205   425 Can't build data connection: Connection refused.
[14.10.0.10] [31.3.3.7] FTP: C PORT=37205   PORT 192,168,0,5,0,80
[31.3.3.7] [14.10.0.10] FTP: R PORT=37205   200 PORT command successful.
[14.10.0.10] [31.3.3.7] FTP: C PORT=37205   LIST
[31.3.3.7] [192.168.0.5] TCP: D=XX S=XX SYN SEQ=4240951199 LEN=0 WIN=65535
[192.168.0.5] [31.3.3.7] TCP: D=XX S=XX SYN ACK=4240951200 SEQ=2193395373 LEN=0 WIN=65535
[31.3.3.7] [192.168.0.5] TCP: D=XX S=XX     ACK=2193395374 WIN<<1=65700
[31.3.3.7] [14.10.0.10] FTP: R PORT=37205   150 Opening ASCII mode data connection for '/bin/ls'.
[31.3.3.7] [14.10.0.10] FTP: R PORT=37205   226 Transfer complete.
[31.3.3.7] [192.168.0.5] FTP: R PORT=XX   Text Data
[14.10.0.10] [31.3.3.7] FTP: C PORT=37205   PORT 192,168,0,5,1,189
[31.3.3.7] [14.10.0.10] FTP: R PORT=37205   200 PORT command successful.
[14.10.0.10] [31.3.3.7] FTP: C PORT=37205   LIST
[31.3.3.7] [192.168.0.5] TCP: D=XX S=XX SYN SEQ=2240251199 LEN=0 WIN=65535
[192.168.0.5] [31.3.3.7] TCP: D=XX S=XX SYN ACK=2240251200 SEQ=4193395373 LEN=0 WIN=65535
[31.3.3.7] [192.168.0.5] TCP: D=XX S=XX     ACK=4193395374 WIN<<1=65700
[31.3.3.7] [14.10.0.10] FTP: R PORT=37205   150 Opening ASCII mode data connection for '/bin/ls'.
[31.3.3.7] [14.10.0.10] FTP: R PORT=37205   226 Transfer complete.
[31.3.3.7] [192.168.0.5] FTP: R PORT=XX   Text Data
```

(a) What is the name of the attack technique used and what are possible mitigations for the attack?

(b) The police can conclude that:

1. The suspect (you) was trying to list the shared folder on host 192.168.0.5.
2. The suspect was trying to probe TCP port 80 on a private host behind 31.3.3.7
3. The suspect found out that TCP port 80 is open on 31.3.3.7
4. The suspect could connect to TCP port 189 of 192.168.0.5.
5. While not shown (for brevity), the same attack can be used to attack any TCP or UDP port.

5. **Web vulnerabilities**

(a) Explain why it is often unfortunate if your name is Sara O'Connor.

You recently applied for membership of the Challenge Liberation Front, a legendary group of hackers of limitless abilities. To prove yourself worthy, you have to change Herbert's password in the CNS administrator's database. You look at the website and see that there is a way to sign up as a student with username and password. There is also a form to change your password later. Of course, the CNS folk are cunning and properly escape all single quotes in the inputs.

In other words, to add a user, they do something like this:

```
sql = "INSERT INTO USERS(uname,passwd) " + "VALUES (" + escape(uname)+ "," + escape(password) +")";
```

To update the password, you have to be logged in already (so users need not provide the username), but you provide the password as (escaped) input, which is used as follows:

```
new_passwd = request.getParameter("new_passwd");
uname = session.getUsername();
sql = "UPDATE USERS SET passwd='"+ escape(new_passwd) + "' WHERE uname='" + uname + "'";
```

(b) Explain how you may still be able to change Herbert's password and impress your new buddies.