7 Assignment (10+2 Points)

Hinweis:	Abgabe in $\{2, 3, 4\}$-er Gruppen.
Abgabe:	**10.06.2017, 23.59**
Email:	Betreff "[Compsec] Ex7'
	(bitte nur .pdf oder .txt, kein .doc, .jpeg, etc)
	Source code: bitte inkl. `signify` Signatur

Exercise 21 (Real world access control matrix **(3 Points)**).
Consider the following file listing (left) taken from a Linux file system and a corresponding `groups` file (right).

```
drwxr-xr-x  alice users .       f:*:1000:frank
drwxr-xr-x  alice users ..      bfct:*:1001:bob,frank,carl,tim
-rwxr--r--  alice f     A       c:*:1002:carl
-rwxrw----  alice bfct  B       ct:*:1003:carl,tim
-rwxr-----  frank c     C       at:*:1004:alice,tim
-rw-r-----  bob   ct    D       other:*:99:
-r-xr-x---  tim   at    E
-r-----r--  carl  c     F
```

Model the permissions indicated by this file listing in an access control matrix using the following access rights:

$own \in P[S, O] \Rightarrow$ subject S owns object O.

$read \in P[S, O] \Rightarrow$ subject S may read object O.

$write \in P[S, O] \Rightarrow$ subject S may write object O.

$exec \in P[S, O] \Rightarrow$ subject S may exec object O.

$inh \in P[S, S'] \Rightarrow$ subject S inherits all rights from subject S'.

Exercise 22 (Case study μ-`shout` (iv): Privilege Dropping (**3 Points**)).
So far μ-`shout` is running as user root. You are worried someone might attack your server successfully and gain root privileges on your machine. Create a new (non-login) user `_ushoutd` and read `SETUID(2)` and `CHROOT(2)`. Adapt your implementation so that your server

1. drops privileges to `_ushoutd` as soon as possible after startup, and
2. is left with a minimal view of the file system (a chroot).

What are the limitations of using `chroot`?

Exercise 23 (More programming mistakes... (**1+2+1 Points + 1 Bonus**))**.**
Consider the following C-program.

```
1  void doAccept(){
2    printf("Password␣Accepted\n");
3    // do sth. else
4  }
5  void doDeny(){
6    printf("Access␣denied\n");
7  }
8  int pwCheck(){
9    char buffer[5];
10   scanf("%s", buffer);
11   return 0;
12 }
13 int main(){
14   if(!pwCheck())
15     doDeny();
16   exit(0);
17 }
```

1. Explain the contents of the stack when the program reaches line 10 (right before scanf is called).
2. Give an input string (in hex/binary) that makes this program print "Password Accepted" on your Debian VM (the program is allowed to crash afterwards), when compiled with *gcc* without further parameters . Explain in detail how you found your solution and why it works.

   Bonus: Adapt your input string so that the program does not crash and explain how it works.
3. Compile this program with `-fno-stack-protector` on your OpenBSD machine. Does your approach work here as well?

Hints: you may want to use `PRINTF(1)` and pipes

```
$ printf '\x20' | ./a.out
```

or `XXD(1)` and IO-redirection (`./a.out < input`) to test your solution.

Exercise 24 (Keeping your systems secure (**Bonus: 1 Points**))**.**
Are there any new vulnerabilities for your Debian or OpenBSD system since last week (03.06.2016 at 23.59)? If so: state one, **name the programming mistake**, decide if you are affected or not, and report if there are any known work-arounds or patches.