5 Assignment 5 (11 Points)

Hinweis:	Abgabe in $\{2, 3, 4\}$-er Gruppen.
Abgabe:	**27.05.2017, 23.59**
Email:	Betreff "[Compsec] Ex5"
	(bitte nur .pdf oder .txt, kein .doc, .jpeg, etc)
	Source code: bitte inkl. `signify` Signatur

Exercise 13 (Rainbow tables (**3 Points**))**.**
Consider the hash function $H : \{0, \dots, 9\}^{2n} \to \{0, \dots, 9\}^2$ with

$$H(x_1 x_2 \dots x_{2n}) = y_1 y_2,$$

where

$$y_1 = \left(\sum_{i=1}^{n} x_{2i-1} \right) \bmod 10 \qquad y_2 = \left(\sum_{i=1}^{n} x_{2_i} \right) \bmod 10$$

(H sums all digits on even (uneven) position modulo 10, e.g. $H(4671) = 17$.).
You are given the following rainbow table with chain length 8.

start point	end point
2345	37
7033	97
4234	79
3400	11
1234	59
7455	71

The regeneration function used in every round [7] is $R : \{0, \dots, 9\}^2 \to \{0, \dots, 9\}^4$ with

$$R(x_1 x_2) = y_1 y_2 y_3 y_4,$$

where $y_1 = x_1, y_2 = x_2$ and

$$y_3 = (x_1 + 3) \bmod 10 \qquad y_4 = (y_2 + 5) \bmod 10.$$

For example: $R(68) = 6893$).

- Find one inverse of the hash value 91 using the table above and document your steps.

[7] to make it less complicated - even though it misses the "rainbow" property this way.

Exercise 14 (x86 assembly recap **(1+2 Points)**).
Recapitulate how machine code is executed on x86 platforms. In particular

1. What are EIP, EBP and ESP registers? Where do they point to?
2. What is an "ABI"? And what does the ABI look like for C programs compiled with gcc on your x86-32 Debian virtual machine?

Exercise 15 (Common mistakes / lessons learned **(4 Points)**).
Consider the following C-program:

```
int main(){
  int authenticated = 0;
  char buffer[9];
  printf("Password: ");
  scanf("%s", buffer);
  if(strcmp(buffer, "password") == 0)
    authenticated = 1;
  if(authenticated){
    printf("Password OK\n");
    // do something else
  }
  else
    printf("Access denied\n");
  return 0;
}
```

Compile and run this program with `gcc` on your Debian virtual machine (the one you installed in the very first exercise).

- Give one password, that does not start with "`password`", but for which the program outputs "`Password OK`" and **does not crash**,
- **explain in detail** why and how it works, and
- why it doesn't work on your OpenBSD virtual machine [8].

Hints: To explain your solution, you may want to use the tool `objdump` to disassemble your program. You may also want to use `gdb` to disassemble the program while it is running or examine it's memory.

Exercise 16 (Keeping your systems secure (**Bonus: 1 Points**)).
Are there any new vulnerabilities for your Debian or OpenBSD system since last week (20.05.2016 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.

[8] Is this intentional or just by accident?