Assignment Sheet #3

Deadline: April 9, 23:59 CET.
Note: Please do not copy text from the Internet, write it yourself, and cite your sources.

Exercise 1.
PGP.
For this homework assignment you need to use PGP. To start using it, you'll need to install an OpenPGP software on your computer. Below you'll find a list of possible solutions for your operating system:
OS X: GPG Keychain (`https://gpgtools.org/#gpgsuite`)
Windows: Kleopatra (`http://gpg4win.de/download.html`)
Linux: GnuPG (`https://www.gnupg.org/download/supported_systems.html`)
iOS: iPGMail (`https://itunes.apple.com/us/app/ipgmail/id430780873?mt=8`)

1. You can find my public key file (file `standash_pub.asc` in the archive `pkey.zip`) on Moodle[1]. Please import the public key into your local OpenPGP Key-Manager.
2. Send me a signed (*with your own signature*) but not encrypted e-mail containing your name and the fingerprint of *my* public key.

Note: I will reply to you that I successfully received your e-mail.

Exercise 2.
For this exercise you can use the ADTool software, or draw an attack tree by hand and scan it.
ADTool: `http://satoss.uni.lu/members/piotr/adtool/`

Consider `https://calendar.google.com/`, an online calendar service.

1. Give an attack tree for the goal "mark a business meeting at 9:00 on April 1st, 2018 for Prof. Mauw" Your attack tree shouldn't be too deep (a depth of about 4 should be enough) and not too wide (similarly, around 4 wide). Furthermore, the attack tree must include at least two "AND" nodes and at least two "OR" nodes.
2. Report the number of unique attacks captured by your tree.
3. Give each leaf of your attack tree a price-tag in euro (how much money does it cost to execute the attack stated in that leaf). Make sure each price is unique.
4. Show the aggregated costs for each node, that is, for each node, determine the cheapest attack (from your tree). What is the cheapest attack on the root node? How much does it cost?

[1] Alternatively, on the webpage `http://satoss.uni.lu/courses/networksecurity/`

Exercise 3.
Web security. Consider a PHP script that receives user input `$user` (the username), `$passwd` (the user's password) and `$comment` (a comment). Suppose these variables are used in PHP to construct the following SQL query. The result of the query is used to determine whether a correct username and password have been supplied: authentication succeeds if the result of the query is non-empty and fails otherwise.

```
$query = "SELECT * FROM users
          WHERE name = '$user' AND password = '$passwd'";
```

1. Give values for `$user` and `$passwd` that make authentication succeed.
2. Give values `$user` and `$passwd` that delete (DROP) the user table from the database.

To avoid SQL injection attacks, `$user` and `$passwd` are made safe as follows:

`$safe_user = filter_var($user, FILTER_SANITIZE_STRING)`, and
`$safe_passwd = filter_var($passwd, FILTER_SANITIZE_STRING)`.

3. Describe (in English) the effect that the function `FILTER_SANITIZE_STRING` must have on the input to prevent SQL injection attacks.

Suppose `$comment` is saved, and later put on a webpage as follows:
`print "<p>$user wrote:<br>$comment </p>";`.

4. What type of attack can be mounted against visitors that view the page. Give the value for `$comment` that needs to be supplied.