

Web Security Lab

Stig F. Mjølunes, Lars L. Ludvigsen,
Karin Sallhammar, Kjetil Haslum, Anton Stolbunov, Md. Abdul Based, Simona Samardjiska

February 11, 2011

0 General Information

0.1 Objectives

This exercise is a mandatory part of the course. Your results will be assessed and counted as 20% of the final grade.

The project focuses on establishing hands-on experience with state-of-the-art security mechanisms in selected web security software: browsers, web servers (Apache), static and dynamic Web pages (PHP), and session/application layer security (SSL). Have a look at the References section for the current software documentation.

The main three assignment parts will be to install and secure:

1. Your own Certificate Authority (CA) based on OpenSSL [1] with Digital certificates signed by the TTM4135 staff.
2. Apache web server [3] with client/server authentication with X.509 certificates access protected HTML pages.
3. Login with PHP [4] script using MySQL [5], cookies and sessions.
4. A Subversion Repository (SVN) [13] [Optional].

Most of the assignment tasks include configuration and programming directly related to relevant security mechanisms, but, as in real engineering, some chores have to be carried out too. The purpose is to experiment with practical configuration of typical web server implementation.

0.2 Groups

This project shall be performed in groups of three students (two might be acceptable in some exceptional cases). The group registration will be part of the exercise (see Section 1.2.4). Consider coordinating your choice of group with projects in other courses running in parallel.

0.3 Laboratory and Supervision

Most of the lab can be done from a computer with Internet access and an SSH client installed. You can use the computers in **Sahara Lab, F-272 Electro building** any time during the lab

weeks (weeks 7-9), as long as you find a free workstation in the lab. See a timetable of parallel courses running in Sahara Lab at <http://www.item.ntnu.no/calendars/sahara.php>.

If you're not a Dep. of Telematics student, you might not have a user account in Sahara Lab. Please check this by trying to log on to one of the computers. If you need a user account, send an email to drift@item.ntnu.no.

If you're stuck with a problem, google it, ask a group next to you, and then ask one of the teaching/student assistants during the assistance hours:

Md. Abdul Based based@item.ntnu.no, office B220.

Simona Samardjiska simonas@item.ntnu.no, office B220.

K.M. Imtiaz-Ud-Din imtiazud@stud.ntnu.no.

Erlend Heimark erlenhei@stud.ntnu.no.

Ingrid Hjulstad ingrihju@stud.ntnu.no.

Assistance in Sahara will be given in the following hours:

Week	Date	Time
7	Tue Feb 15	11:15-14:00
7	Thu Feb 17	14:15-17:00
7	Fri Feb 18	9:15-12:00
8	Tue Feb 22	11:15-14:00
8	Thu Feb 24	14:15-17:00
8	Fri Feb 25	9:15-12:00
9	Tue Mar 1	11:15-14:00
9	Thu Mar 3	14:15-17:00
9	Fri Mar 4	9:15-12:00

0.4 Set Up

During the lab you will access three servers:

login.stud.ntnu.no. Access this server over SSH with your student user account and password.

You will generate your personal certificate request and later a .p12 file for your browser here. See Section 1 for more details. In fact, these tasks can be accomplished on any PC with OpenSSL installed. As soon as you get your group password you should do the rest of the exercise on the **ttm4135.item.ntnu.no** server.

course4135.item.ntnu.no. Access this server over https. You will register your email, upload your personal certificate request, download a signed certificate, register in a group, get password for **ttm4135.item.ntnu.no** server, upload your group CA certificate request and download a signed certificate for the group CA. See Section 1 for more details.

This server runs Apache with SSL and PHP. It also runs a PostgreSQL database to store your personal information (name, email, sha1(password), certificate request, certificate), and information about the groups (group number, password, certificate request, certificate). The address for the main page is: <https://course4135.item.ntnu.no>. Only those students that have registered their email address may start working with the laboratory exercise. If you have forgotten your personal password you can repeat the email registration procedure and you will be able to choose a new password.

ttm4135.item.ntnu.no. Access this server over SSH with your group account and password. Your user account on the server will be grnn, where nn is your group number (e.g. gr03, gr12). Details on getting the group password will be given in Section 1.2.4. On this server you will configure your group CA, generate group CA's certificate request, install and configure Apache web server with SSL, and create some PHP pages. Groups should not install their web servers on any of their own machines, but rather use remote access to this server. See Sections 2, 3 for more details.

The PCs in Sahara provide Secure Shell Client (SSH) program to connect to the servers.

0.5 Laboratory Journal (log)

It is best practice to keep a journal of the complete work process, to base your laboratory report on. One good reason for doing the journal is that you have a limited time available in the lab, so your notes will be of great use when composing the report outside the lab hours. Maintaining a journal is also a proper engineering and scientific approach to your work.

As soon as the web server is up and running, you should make the journal available on the web to the teaching assistants. The laboratory journal will not be assessed by the staff. However, it can be used if we have some questions regarding your work.

The recommended structure for the laboratory journal is the following. Each section of the journal should describe one work session (for example one day's work) and include a description of the work according to the following template:

- A heading that includes the date, place, duration of work and the participating group members.
- Objectives. A few sentences on the current problem and the primary goals for the day.
- A preliminary plan for the work.
- Foundation and references necessary for the work.
- The course of actions and procedures: Include notes on how you performed your work in terms of a step-by-step description. Describe the components and configurations that were used, so that you will be able to explain to others what you have done.
- An overview over results, observations and gathered data. Present the results, data and experiences, preferably in well-arranged tables and figures.
- Discussion and conclusion: Explain the results and the reason for them. Were the primary goals fulfilled? What should have been done different, and why? Plan your subsequent work and identify the following activities.

The journal shall be published on an access-protected web page under the group's Apache server.

0.6 Acceptance and Evaluation

The assignment submission consists of two parts:

1. Demonstration of achieved milestones (see below).
2. Lab report.

0.6.1 The Milestones

We have set a few milestones for you, where we will check your work. Make sure you have shown your work results for each of the milestones to one of the teaching assistants. The approval of the milestones is on the pass-fail basis.

Part I Section 1.2. Post your personal certificate request before Monday, Feb 14.

Part I. Certificate Authority Section 1 should be done before the end of week 7 (Feb 18).

Part II. Apache Section 2 should be done before the end of week 8 (Feb 25).

Part III. PHP Section 3 should be done before the end of week 9 (Mar 4).

Part IV. A Subversion Repository Section 4 should be done before the end of week 9 (Mar 4) [Optional].

0.6.2 The Report

Your report will be assessed and count as 20% of the final grade. The report must be submitted via It's learning within Friday, Mar 11, 23:59:00. Remember to add all group members on the form when uploading the report to it's learning.

A laboratory report is a structured document, usually written and polished after the work is completed. The report is based on notes recorded during the course of experimentation (laboratory journal). The focus of the laboratory is to gain an understanding of how state-of-the-art web security can be accomplished using freely available open source tools. It is important to focus on security weaknesses and strengths of the implemented solution. The recommended structure of the report is: Title Page (or heading on the first page), Introduction, Experimental Procedure (optional, if present then only describes occasions when you did not follow the lab description procedure), Results, Discussion, Conclusion, References, Appendices. See [7] for a more thorough description. You should also answer the questions marked with **Q** in this lab description.

Important evaluation criteria are:

- Format
 - The report is limited to 4 A4 pages including text, references and figures, but exclusive of the title page and appendices.
 - Use 11pt font size, normal line separation, one-column layout and reasonable margins.
 - The submitted file format must be pdf.
 - You should write either in English or in Norwegian, with good grammar and syntax.
 - Title page (or heading on the first page) should include names of all group members, the group number, date and title.
 - The reference list should be formatted according to the common rules for citation [6].
 - We recommend using L^AT_EX when writing your report.
- Content
 - Clear and logical structure.
 - Clear identification of problems and objectives.
 - Precision of facts.

- Presentation, your own analysis and evaluation of the results.
- Logical discussion part with reasoning that clearly shows that you have understood what you have been doing.
- Answers to the **Q**-questions (include question numbers and text when answering).
- Quality of references. As a starting point you could refer to the course textbooks, but other references are required as well. When you find information on the Internet, it is important that you are careful about its quality. We recommend that you search for information in the university library databases (BIBSYS, eUBiT søkeportal and other databases), and that you choose references to published articles and books.
- Your level of understanding of the material.
- Presence of new ideas.

0.6.3 Creativity

The assignment is formulated as a set of minimum requirements. We encourage you to let your creative engineering power take you beyond the path we have been staking! For example, you may challenge other groups in terms of their “implementation security”, using analysis and different kinds of attacks. However remember that your provocative activity should not hinder other people’s work. On the other hand, if your web site happens to be “hacked”, try to analyse reasons and implement the necessary countermeasures. It is wise to demonstrate such security incidents to the teaching assistants in order to get additional points.

0.7 Useful Unix Commands (optional)

To get a soft start, we start with an introductory part of the Linux environment. If all of the members of your group have satisfactory experience with Linux/Unix commands, you may skip this part.

Note that editors installed on **ttn4135.item.ntnu.no** server are vi, vim, emacs and nano. Of course, it is also possible to edit the configuration files on the client PC and then upload them to the server.

Start by figuring out how to contact the **login.stud.ntnu.no** server (or **ttn4135.item.ntnu.no** if you’ve already got the group password) by using SSH. Then try some of the shell commands of Linux. If you need more information on a certain command, you can type **man <command>** to get help on OS commands and utilities. For instance, try mandatory

```
$ man ls
```

Here are 10 more commands that might be useful in your further work.

chmod - **change file access permissions**. Create a subdirectory (mkdir) in your home directory and put a file welcome.txt with a short message in this subdirectory. Set the permission bits on the subdirectory so that the owner has execute access. Try to:

- make the subdirectory the current directory with cd
- list the subdirectory
- display the contents of welcome.txt
- create a copy of welcome.txt in the subdirectory

Repeat the same experiments first with read permission and then with write permission on the subdirectory.

cat - concatenate files and print on the standard output. Create a new file goodbye.txt with a short message in the same subdirectory as welcome.txt. What does the command:

```
$ cat welcome.txt goodbye.txt > readme.txt
```

cp - copy files and directories. Use this command to copy a file, and then to copy a directory.

mv - move (rename) files. Use this file to move a file, and to rename a file.

passwd - change user password. What are the requirements of Unix passwords? What are two reasons for using the passwd command?

rm - remove files or directories. How do you remove directories with the rm command?

tail - output the last part of files. What does the command tail do? When might this be useful in our project?

wget - non-interactive network downloader. Find a file with md5 fingerprint and download it with wget. In order to work with SSL connections consider the option `-ca-certificate`.

md5sum - compute and check MD5 message digest. Check the md5 fingerprint of the downloaded file with md5sum.

scp - secure copy (remote file copy program). Simplified syntax, see the man page for more information:

```
$ scp user@host1:file1 user@host2:file2
```

1 Part I: Certificate Authority

This part of the project should be finished by the end of week 7.

1.1 Understanding certificates

The purpose of this part is to set up and configure a certificate authority (CA) called "Grnn CA" (e.g. "Gr03 CA" etc.) For this, you will use the OpenSSL software. Use this application to create the certificate signing requests (CSR), and for signing certificates. Figure 1 displays the certificate hierarchy environment of this project. A comprehensive introduction to SSL and openssl is available at [2]. You can, to a large extent, follow this receipt with one important exception: this tutorial will generate a root certificate, which is self-signed. You are not going to use a self-signed certificate in this assignment, rather you must generate a CSR and upload it onto the course4135 server. In return, your public key will be certified.

1.2 Generate your Personal Certificate

To continue the lab you need to create and post a certificate request for your personal certificate. When it is signed by the staff, you have to create a .p12 file that contains your personal certificate and your private key, this file is then imported into your browser. This part of the exercise should be done by login at **login.stud.ntnu.no** with SSH. It can also be done at a PC with installed OpenSSL.

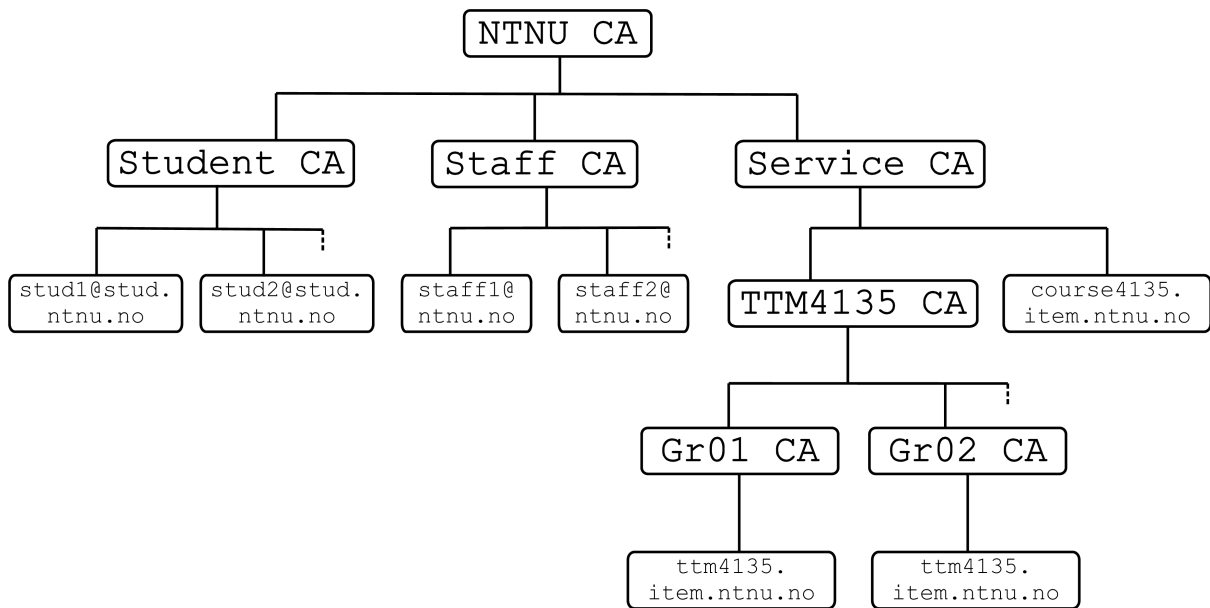


Figure 1: The certificate hierarchy used in the course TTM4135.

1.2.1 Create a certificate request for your student certificate

An open source tool called OpenSSL is used for creating certificate request. OpenSSL is installed on most Linux servers at NTNU, but you can also download and install it on your own computer. For a complete documentation of the OpenSSL tool, see [1].

- At first register your email address on the web: <https://course4135.item.ntnu.no/>
- In order to make the certificate request you have to download the configuration file for OpenSSL: https://course4135.item.ntnu.no/open/stud_openssl.cnf. When downloading with wget, you'll need the certificate file ntnuca.pem, which is put on it's learning in the CACertificate folder. The configuration file stud_openssl.cnf should be stored in the same directory as used when executing the openssl command.
- Then you have to execute the following command on a Linux box to generate the certificate request. Use the default value for all fields except the Common Name field, which has to be your email address registered on course4135 server:

```
$ openssl req -new -nodes -keyout stud_private.key \
-out stud_cert_req.pem -config ./stud_openssl.cnf
```

- This command will generate a private key stud_private.key and a certificate request stud_cert_req.pem. It is very important to keep the private key secret to avoid impersonation.

More details on openssl req can be found at [1, req].

1.2.2 Upload the certificate request

The certificate request have to be signed by the TTM4135 staff. Follow the certificate request upload page: https://course4135.item.ntnu.no/open/stud_cert_post.php.

The certificate request will be inspected by the TTM4135 staff and signed if everything is correct. You will then (after some time) receive an email containing a link to your certificate or a message explaining what is wrong with the request.

1.2.3 Create a .p12 file

When you have received the email containing the signed certificate you have to create a .p12 file that can be imported by the browser. Download your student certificate here: https://course4135.item.ntnu.no/open/get_stud_cert_form.php.

- The .p12 file have to be created by the OpenSSL tool.
- You need to specify the following arguments to make the .p12 file.
 - in The certificate file signed by the TTM4135 staff: email.pem.
 - inkey Your private key file created in the first step: stud_private.key.
 - name The name that will appear in the browser: username.
 - out The name of the .p12 file: email.p12.
 - certfile The certificate chain file for the Student CA: https://course4135.item.ntnu.no/open/studca_chain.pem.
- To generate the .p12 file execute:

```
$ openssl pkcs12 -export -in email.pem -inkey stud_private.key \
-name your_username -out email.p12 -certfile studca_chain.pem
```
- Open ssl will ask for an “Export Password”, you will be asked for this password when you import the .p12 file into the browser.

More details on openssl pkcs12 can be found at [1, pkcs12]..

1.2.4 Register a group

To get access to the group registration page you need to import the student certificate (.p12 file) into your browser.

For IE 6: [IE 6 certificate import](#).

For Firefox:

1. Choose: Tools→Options (or Edit→Preferences).
2. Choose: Advanced, and Tab: Security.
3. Click: View Certificates.
4. Click: Import and select your .p12 file.

Then, follow this link to register a group for the lab: <https://course4135.item.ntnu.no/stud/group.php>. Only students with a valid personal certificate in their browser will have access to the group registration and group-certificate management pages, so before you continue the exercise make sure that you have:

- installed your personal certificate in your browser;
- registered to a group.

Now you can logon to the **ttm4135.item.ntnu.no** server with your group’s username and password.

1.3 Generating a Group Certificate Request

This should be done on the `ttm4135.item.ntnu.no` server.

Make a CSR for the group CA (use Common Name = Grnn CA), where nn is the group number assigned to you. The number should be stated with two numbers, i.e. 'Gr01 CA' for the first group, etc. You can use the configuration file that you used when you created the student CSR. To check that your certificate signing request is correct, you can type

```
$ openssl req -in grnn_req.pem -verify -text -noout
```

where `grnn_req.pem` should be the name of the file containing your groups CSR. Upload your group CSR on the `course4135` server. It may take a little while before a teaching assistant manually certifies your CSR. You will receive an email when the certificate is ready to be downloaded. If you experience too long waiting time, contact a teaching assistant to speed up the process. Of course, you may start with Part II (installation of the Apache web server) while waiting for the certificate.

To show the content of the certificate try the following commands:

```
$ cat cacert.pem
$ openssl x509 -in cacert.pem
$ openssl x509 -in cacert.pem -noout -text
$ openssl x509 -in cacert.pem -noout -fingerprint
```

1.4 Setting up the Group CA

Now when you have a signed group certificate, you can set up the group CA. We suggest that you use the following directory and file structure for the group CA:

```
/home/grnn/ca (dir)
/home/grnn/ca/private (dir)
/home/grnn/ca/private/newcerts (dir)
/home/grnn/ca/private/serial (file)
/home/grnn/ca/private/index.txt (file)
/home/grnn/ca/private/caconf.cnf (file)
/home/grnn/ca/private/cacert.pem (file)
/home/grnn/ca/private/cakey.key (file)
```

Create the configuration file `caconf.cnf` for the CA, so it match the above file and directory structure. For more information on setting up a CA see [2]. Remember that this tutorial is getting old, hence some of the cryptographic variables in the configuration file might need to be updated. Please explain in the report which variables you have changed and why. Also note that to create the correct format of your server's certificate, it is important that the following lines are included in your OpenSSL configuration file:

```
[ server_cert ] # to sign the Apache CSR
basicConstraints      = critical, CA:FALSE
subjectKeyIdentifier  = hash
keyUsage              = digitalSignature, keyEncipherment
nsCertType            = server
```

1.5 Generate a Certificate for your Apache Web Server

Now you are ready to use your group CA to create a signed certificate and the corresponding private key for your Apache web server. Use the option [server_cert] to sign it. Be careful about which “Common Name” you choose, so that your Apache server will be able to authenticate itself with the certificate!

Hint: Do not install this certificate in any of your browsers! Rather store the files in an appropriate subdirectory at the ttm4135 server, i.e.

```
/home/grnn/apache/servercert/server_cert.pem  
/home/grnn/apache/servercert/server_key.pem
```

Q1. Comment on security related issues regarding the cryptographic algorithms used to generate and sign your groups web server certificate (key length, algorithm, etc.).

2 Part II: Access Control and Apache

This part of the project should be finished by the end of week 8.

2.1 Source Files and their Authenticity

Download the source files for the newest Apache webserver for Linux (apache 2.2.17) and the corresponding Pretty Good Privacy (PGP) detached signature (it is assumed that you are logged on to the **ttm4135.item.ntnu.no** server):

```
$ wget http://www.powertech.no/apache/dist/httpd/httpd-2.2.17.tar.gz  
$ wget http://www.apache.org/dist/httpd/httpd-2.2.17.tar.gz.asc
```

Check the PGP detached signature of the Apache source code using GnuPG (see Exercise 1 for the TTM4135 course which is put on it’s learning in the Exercise folder). Also check the fingerprint of the signer’s public key. See [8] for more information.

Q2. Explain what you have achieved through each of these verifications. What is the name of the person signing the Apache release?

2.2 Compiling and Installing

Start by compiling Apache. Extract the file by using the commands:

```
$ gzip -d httpd-2.2.17.tar.gz  
$ tar xvf httpd-2.2.17.tar
```

Change to the new subdirectory in your home directory containing the Apache source code. Now set your the Apache directory and include the SSL module in your configuration:

- Specify the directory where Apache should be installed:
–prefix=/home/grnn/apache.
- Enable mod_ssl: –enable-ssl.
- Enable mod_so: –enable-module=so (used to load PHP).

```
$ mkdir apache
$ cd httpd-2.2.17
$ ./configure --prefix=/home/grnn/apache \
--enable-ssl --enable-module=so
```

It is crucial that you configure with the correct prefix, otherwise you will not be able to install the web server when the compilation is ready. Now compile and finish the installation:

```
$ make
$ make install
```

If you experience compilation problems, consult the Apache documentation [3, Compiling and Installing].

2.3 Configure httpd

Web server configuration time is next. Change current directory to the directory of your Apache server installation. Take a look at the configuration file `apache/conf/httpd.conf` and make the necessary changes with respect to port number (hint: find the line "Listen 80").

2.3.1 Server Port

The server processes shall run under the group's user on the server. You must use your own dedicated ports on the server. Each group is assigned 10 ports, which should be more than enough for this purpose. The first port number for group n will be $8100 + 10 * (n - 1)$. Hence, group number 1 can use any of the port numbers 8100 – 8109. Group 5 will then begin on 8140 etc. Usage of ports outside your range may interfere with the other group's servers and cannot be tolerated. Choose the first assigned port value for your HTTP server. Thus for Group 5 the Apache server configured to use port 8140 should make the initial web page available on the URL `http://ttm4135.item.ntnu.no:8140`.

Verify the server installation by (re)starting Apache

```
$ apache/bin/apachectl stop
$ apache/bin/apachectl start
```

and test the installation with a browser (e.g. by browsing `http://ttm4135.item.ntnu.no:8140/`).

2.4 Virtual Host

The virtual host directive is used when more than one web server is installed on the same machine. In this exercise each group will install one HTTP server and one HTTPS server at the `ttm4135.item.no` machine. The two servers are separated by their port numbers. Firstly make one virtual host section for the HTTP server, and later you have to add a new virtual host for the SSL server. All configurations specific for the HTTP server should be placed inside the virtual host section for the HTTP server. For more details see [3, Virtual Hosts].

In order to make it easier for the staff to help the students, we suggest that you do all the configurations for the web server in a new file called `group.conf` located in the `/home/grnn/apache/conf` directory. The only thing you need to do in the `httpd.conf` file is to include the `group.conf` file. At the end of `httpd.conf` add the following line:

```
Include conf/group.conf
```

Create the `group.conf` file, it should be similar to the following example:

```
NameVirtualHost *:8140
<VirtualHost *:8140>
ServerAdmin myemail@stud.ntnu.no
DocumentRoot /home/grnn/apache/htdocs
ServerName ttm4135.item.ntnu.no
<Directory /home/grnn/apache/htdocs>
Allow from all
</Directory>
</VirtualHost>
```

Test your new configuration by restarting Apache. Consider including `Options -Indexes` to disable automatic listing of the directory contents for security reasons.

2.5 Configure SSL

Finally, we enter into security setup work. We need to change configurations to secure the web server. Documentation on how to setup and run SSL on Apache can be found at [3, SSL/TLS Encryption].

All configuration parameters are set in the `group.conf` file, you have to make a new virtual host for the SSL server. Remember to add a new `Listen` command for the https server, this should be done immediately before the new virtual host section. There are already one `Listen` directive in the `httpd.conf` file that is used by the HTTP server. You should now have two virtual host sections in the `group.conf` file, one for the HTTP server and one for the SSL server. The following parameters and options must be set to correct values:

Listen - the port number for the SSL connections on Apache.

SSLEngine - to enable SSL.

Servername - the URL (and port number) for the SSL on Apache.

SSLCertificateFile - the path to the Apache server certificate.

SSLCertificateKeyFile - the path to the Apache server private key.

SSLCertificateChainFile - the path to the Apache server certificate chain.

Choose the group's port number for HTTPS (of course, this must be different from the port number used for the plain HTTP). Recall your private key and the SSL certificate for the Apache web server (in Part I) that you generated by using the group CA. You can create the web server's certificate chain by concatenating the group CA certificate, the TTM4135 CA certificate, the Service CA certificate, and the NTNU CA certificate (see Figure 1). These certificates can be downloaded here: <https://course4135.item.ntnu.no/open/certs/>.

Note: Restart the Apache server to see the result of any changes made to the configuration files `httpd.conf` and `group.conf`.

The web page [10] provides more information on how to configure SSL. Please note that most of this documentation is written for older versions of Apache and `mod_ssl`, hence parts of it might be out of date.

2.6 Client Authentication

Your task is to give the correct permissions to the directories listed in Section 2.7.1. Therefore, you must implement client authentication on your Apache web server. The authentication procedure must rely on verifying digital certificates. Information on how to set up SSL for client authentication under Apache 2.0 is available at [9] (read the section “How can I allow only clients who have certificates to access a particular URL, but allow all clients to access the rest of the server?”).

To authenticate users on your Apache web server by means of digital certificates and secure protocols, the `StdEnvVars` primitive should be used. The variables `SSLCACertificateFile` and `SSLRequire` need to be set in the `group.conf` file. See the `httpd-ssl.conf` file in the `conf/extras` folder to see examples in addition to the `ssl_howto` mentioned before.

Note: Consider the possible consequences of your choices before you implement your solution! Remember that possible authentication mechanisms can be either `SubjectDN` or `SerialNumber` together with `IssuerDN`.

2.7 Functional Requirements

The teaching assistants will evaluate your work in part II by checking the following requirements:

1. Your web server is up and running.
2. All clients can connect to the web server by using the HTTP protocol.
3. All clients can connect to the web server by using the HTTPS protocol.
4. The web server can authenticate itself by means of a complete and valid certificate chain.
5. Only the teaching assistants and group members have access to the `restricted` folder, and the lab journal is published in this folder.
6. Only those clients who can present a certificate signed by 'NTNU CA' are allowed to access the `admin` folder.

2.7.1 Directories

`/grnn/apache/htdocs/secure` - this directory should be the root directory of the HTTPS server. It should be accessible to everyone using HTTPS, and forbidden for those using HTTP.

`/grnn/apache/htdocs/secure/restricted` - the directory `restricted` is a sub-directory of `secure`. It is the directory which only the group members and the staff have access to via client certificate authentication.

`/grnn/apache/htdocs/secure/admin` - the directory `admin` is a sub-directory of `secure`, and must be set up to demand client certificate authentication. It should be accessible to all clients with a valid “NTNU CA”-issued certificate. Don't be confused by the directory name, you'll limit the access further in Part III.

Q3. What are the access permissions to your web server's configuration files, server certificate and the corresponding private key? Comment on possible attacks to your web server due to inappropriate file permissions.

Q4. Web servers offering weak cryptography are subject to several attacks. What kind of attacks are feasible? How did you configure your server to prevent such attacks?

3 Part III: Writing PHP Application

This part of the project should be finished by the end of week 9.

You will write a web site in PHP that uses both username/password and X.509 certificates to validate clients. If you haven't programmed in PHP before it is highly advisable to read the tutorial [11] (not all the 15 chapters are relevant to this lab, but in addition to the first introductory chapters, 8 and 10 should be read carefully). A PHP manual can be found at [4].

3.1 Install PHP

In this section you will be guided through compilation and installation of PHP. It will be installed as an Apache module. More details on installing PHP can be found at [4, Installation on Unix systems].

disable command line interface: `--disable-cli`

disable cgi: `--disable-cgi`

set the root for the installation: `--prefix=/home/grnn/php`

enable mysql: `--with-mysql=/usr/bin/mysql_config`

To configure and install PHP:

```
$ gunzip php-5.3.1.tar.gz
$ tar -xvf php-5.3.1.tar
$ mkdir php
$ cd php-5.3.1
```

Then copy the configuration file for PHP from the installation directory into the Apache configuration directory:

```
$ cp ~/php-5.3.1/php.ini-development ~/apache/conf/php.ini

$ ./configure --with-apxs2=/home/grnn/apache/bin/apxs \
  --with-config-file-path=/home/grnn/apache/conf/php.ini \
  --disable-cgi --disable-cli --prefix=/home/grnn/php \
  --with-mysql=/usr/bin/mysql_config
$ make
$ make install
```

Add the following line to the end of the Apache configuration file to tell Apache to execute files ending with .php with PHP (apache/conf/httpd.conf):

AddType application/x-httpd-php .php

3.2 Test Apache with PHP

Make a file `test.php` in the `apache/htdocs/secure/restricted` directory consisting of only one line:

```
<?php phpinfo();?>
```

Then restart Apache. The information returned by this page is useful when developing PHP applications. It is important to notice however that the content also is exploitable by attackers. Do not provide unauthorized viewers access to this data. These lines of code should be removed from the final production environment.

3.3 MySQL Database

There has been set up a MySQL database `dbgrnn` for each group at `ttm4135.item.ntnu.no` where the username and password are the same as for the group's SSH-login (the initial group password that you have received by email). From the command line you can connect to the database by typing:

```
$ mysql dbgrnn -p -u grnn (e.g. mysql dbgr01 -p -u gr01)
```

PHP is compiled with the MySQL Improved Extension (`-with-mysqli...`), which allows access to the functionality provided by MySQL 4.1 and above. When programming the PHP application remember to use the `mysqli` extensions [12].

We will need a database of legitimate clients of our server. Start by creating a table which includes user names and passwords for legitimate clients. Keep in mind that it might not be a good idea to store the passwords in cleartext, and also that using an index as a primary key enhances security.

3.4 Functional Requirements

We will make a sign up page in the `admin` folder. The sign up page will act as an administrative interface to the database of clients. Here you can view the list of all users, add and remove users. Hence it is very important that the Part II of the lab with client certificate validation is done properly, so that only NTNU members can access this area. We'll restrict the access further by means of PHP, making the page accessible only to the group members and staff.

We will also create a log in page which will be accessible to everyone (as long as it is using the HTTPS protocol). The log in page needs to contain a username/password form which can validate clients and successfully send authorized clients to a page with some valuable information. On this page the lab report must be made available for the staff to download.

The hierarchy of files will be the following:

```
~/apache/htdocs/index.html
~/apache/htdocs/secure/index.html
~/apache/htdocs/secure/login.php
~/apache/htdocs/secure/admin/signup.php
~/apache/htdocs/secure/restricted/credentials.html
~/apache/htdocs/secure/restricted/test.php
```

3.4.1 Files

index.html This file should contain links pointing to all relevant pages for the staff, including your laboratory logs.

credentials.html A valid username and password must be generated for the staff and put here so we can log on to the `login.php`-page and download the lab report. Remember to double-check that only the group members and staff have access to this area of the web server!

signup.php This script must be programmed to inspect the client certificate of the web browser, and display two different pages based on whether the client is member of your group/course staff, or if it is only a regular NTNU student. As described earlier, authorized clients should be able to see a table of user names that are already registered in the database, and the possibility to add and remove users from this database. Students with

a valid NTNU-certificate, but not part of the group should encounter a page which tells them they do not have sufficient privileges to enter.

Hint: To be able to distinguish client certificates using PHP, copy the file `test.php` you created earlier to the admin directory. All variables listed under “PHP variable” located on the bottom of the PHP info-page can be used in the PHP-code. For instance the `$_SERVER[“SSL_PROTOCOL”]` variable which should yield `TLSv1`, can be checked by entering this in PHP:

```
<?php
if ($_SERVER['SSL_PROTOCOL'] != TLSv1){
    echo 'Weak security';
} else
    echo 'Security is fine';
?>
```

login.php This page should be accessible to all clients. The script should display three different web-pages depending on inputs: one standard Log in-page when no user input has been forwarded by the log in-form, another page when the Log in is successful, and an error page if the Log in-attempt fails. Users with registered user names and passwords in the database should be able to log in here and get access to the group’s lab report. Store the report in a directory not accessible from the web. Use the PHP file function to read the document and send it to the browser.

3.4.2 Cookie

We want to make it easier for users that use the `login.php` page frequently by creating a cookie. Each time a user enters the page, the user name will appear automatically, so that the user only needs to enter the password. A cookie can store much more information than just a user name. Feel free to experiment with additional info the user can be made aware of when he enters the page.

3.4.3 Session

This functional requirement is not mandatory for the approval of part III.

The web is stateless in the sense that it does not remember which page a user is watching, or if that particular user has been watching other pages on the web site. Without this information it is difficult to create a useful, interactive experience. To remedy this problem, sessions have been invented. They help both the client and the server to maintain a memory of the client’s credentials. A good and easy example of this state information is when a client logs on to a web server. After a client has logged in, the server still needs to know that the client is a trusted user when he requests another page. If not, the client would have to log in again each time he wants to read a restricted page.

To be able to keep track of whether users have logged in to `login.php` successfully or not, a unique PHP session must be started for each log in instance.

Note: to avoid collisions between sessions set by different groups, try to change `session.name` and `session.save_path` variables in `php.ini`. If you ever experience problems with configuring `php.ini`, note that the default path might be `/home/grxx/apache/conf/php.ini/php.ini`.

3.5 Important

Whenever user interactivity is possible, securing your code becomes crucial. Even though most users will use the pages as they are intended, evil doers are always lurking (especially other groups in TTM4135 trying to get a better grade than you). If you do not use input validation in your PHP-scripts, chances are that your database soon will be deleted or tampered with. Be aware of SQL-injection attacks, Cross-site scripting attacks, Session hijacking and other attacks. If you have no clue what this is, google it! Thankfully PHP has some built-in functions to check for unusual characters: `mysqli_real_escape_string()` and `strip_tags()` are two, search the web for more. In addition to these functions you should also do manual checks on inputs to see if they are empty, the length of the input-strings etc.

Q5. What kind of malicious attacks is your web application (PHP) vulnerable to? Describe them briefly, and point out what countermeasures you have developed in your code to prevent such attacks.

4 Part IV: Setting Up A Subversion Repository

This part of the project should be finished by the end of week 9.

In this part you will set up a repository using apache webserver. This part is not mandatory. The first four groups will get 5 bonus points if this part is done successfully. You must approve your results if you finish this part and include the results in your lab report as appendix.

4.1 Compiling Subversion and its Dependencies

First of all, download the latest version of the source code for subversion and its dependencies from <http://subversion.tigris.org/>.

Some of the dependencies needed are APACHE, APR, APR-UTIL, NEON. We assume here that Apache is already installed.

Extract both the subversion source code and the subversion dependencies source code.

```
tar -jxvf subversion-x.x.x.tar.bz2
tar -jxvf subversion-deps-x.x.x.tar.bz2
cd subversion-x.x.x
```

Compile the APR first.

```
cd apr
./configure --prefix=/home/grnn/apr
make
make install
cd ..
```

Next compile APR-UTIL.

```
cd apr-util
./configure --prefix=/home/grnn/apr/arp-util --with-apr=/home/grnn/apr/
make
make install
cd ..
```

And then compile NEON.

```
cd neon
./configure --prefix=/home/grnn/neon
make
make install
cd ..
```

Finally compile subversion with the support for all that was just installed.

```
./configure --prefix=/home/grnn/subversion --with-apxs2=/home/grnn/apache/bin/apxs
--with-apr=/home/grnn/apr/ --with-apr-util=/home/grnn/apr-util/
--with-neon=/home/grnn/neon/ --with-ssl
make
make install
```

We suppose that the configuration file `httpd.conf` has been modified in Part II, and there is a valid user/group to run `httpd` as. If not, add a user, set ownership on the files where apache is installed and modify `httpd.conf` accordingly. (We suppose here that there is a user `apache`.)

4.2 Creating a Repository and Configuring Apache to Serve the Repository

Suppose you want to create a Repository at `/home/grnn/subversion/repository` that uses an FSFS database (as opposed to Berkeley DB database), so execute the command:

```
mkdir -v /home/grnn/subversion/
svnadmin create --fs-type fsfs /home/grnn/subversion/repository
```

That should create a subversion repository under `/home/grnn/subversion/repository`. Take a look of the content of the created repository. You will need it later.

Make sure that the modules `mod_dav` and `mod_dav_svn` are loaded in the apache configuration file and are also present under modules directory.

```
LoadModule dav_module      modules/mod_dav.so
LoadModule dav_svn_module  modules/mod_dav_svn.so
```

Add the following lines to `httpd.conf`.

```
<Location /subversion>
    DAV svn
    SVNPath /home/grnn/subversion/repository/
</Location>
```

4.3 Setting up Authentication

For the authentication you will need to make more changes to the apache configuration file.

```
<Location /subversion>
    DAV svn
    SVNPath /home/grnn/subversion/repository/
    AuthType Basic
    AuthName "Subversion repository"
    AuthUserFile /home/grnn/subversion/repository/conf/svn-auth-file
    Require valid-user
</Location>
```

You should create the `svn-auth-file` using `htpasswd` and add users (with passwords). For further explanation of the options for `htpasswd` see the Apache online documentation.

4.4 Creating Initial Repository Layout, and Importing Initial Data

A repository mostly contains 3 standard folders: branches, tags, trunk.

For creating those standard folders in a repository, create a temporary folder anywhere you want for ex. /tmp with these subdirectories.

```
mkdir -pv /tmp/subversion-layout/{branches,tags}
```

After you have made all the layout folders, move all the contents of your project to the trunk folder.

```
mv -v /home/grnn/apache2/htdocs /tmp/subversion-layout/trunk
```

Then make an initial import of the temporary created directory.

```
svn import /tmp/subversion-layout/ http://127.0.0.1/subversion/
```

This will setup a default repository layout, and make a first revision. Delete the temporary folders since all the files are already in the repository.

4.5 Setting up a Working Copy

Next, you need to make a working copy of all the files in the repository under /htdocs. So that whenever a developer updates the php codes, they can see the code changes taking effect in a working environment.

But setting up a working copy would not accomplish this task, you need to make the hook scripts to work with a working copy. Thus, whenever a developer commits to the repository, the hook script will run itself, and update the working copy.

To setup a working copy, transfer the content of the repository to htdocs/svngrnn.

```
cd /home/grnn/apache2/  
su apache
```

```
/home/grnn/subversion/bin/svn checkout\  
http://127.0.0.1/subversion/trunk/ htdocs/svngrnn
```

Next, you will set up a post-commit hook script. In short, a hook is a program triggered by some repository event, such as the creation of a new revision or the modification of an unversioned property. For more info see svn.apache.org.

Copy the post-commit.tmpl file into post-commit in the same hooks directory, and give post-commit execution rights.

```
cp -v /home/grnn/subversion/repository/hooks/post-commit.tmpl \  
/home/grnn/subversion/repository/hooks/post-commit
```

```
chmod +x /home/grnn/subversion/repository/hooks/post-commit
```

Now edit the post-commit script to look something like this.

```
REPOS="$1"  
REV="$2"
```

```
#/usr/share/subversion/hook-scripts/commit-email.pl \
```

```
# "$REPOS" "$REV" commit-watchers@example.org

sudo svn update /home/grnn/apache2/htdocs/svngrnn/ >> \
/home/grnn/subversion/repository/logs/post-commit.log
```

Note that you should create the post-commit.log file.

Make sure the repository folder has the proper user ownership, it is advised to set ownership on /home/grnn/subversion/repository/ for user apache.

```
chown -Rv apache.apache /home/grnn/subversion/repository/
```

Start the apache server, and test your newly created svn.

References

- [1] OpenSSL. <http://www.openssl.org/docs/apps/openssl.html>.
- [2] Eclectica: Creating and Using SSL Certificates. <http://www.eclectica.ca/howto/ssl-cert-howto.php>.
- [3] Apache HTTP Server Version 2.2 Documentation. <http://httpd.apache.org/docs/2.2/>.
- [4] PHP Manual. <http://www.php.net/manual/en/>.
- [5] MySQL 5.0 Documentation. <http://dev.mysql.com/doc/refman/5.0/en/>.
- [6] Citation. <http://en.wikipedia.org/wiki/Citation>.
- [7] Handbook: Laboratory Reports. <http://www.ecf.utoronto.ca/~writing/handbook-lab.html>.
- [8] Verifying Apache HTTP Server Releases — The Apache HTTP Server Project. <http://httpd.apache.org/dev/verification.html>.
- [9] SSL/TLS Strong Encryption: How-To — Apache HTTP Server. http://httpd.apache.org/docs/2.2/ssl/ssl_howto.html.
- [10] mod_ssl version 2.8 User Manual. <http://www.modssl.org/docs/2.8/>.
- [11] PHP 101: PHP For the Absolute Beginner. <http://devzone.zend.com/node/view/id/627>.
- [12] PHP: Mysqli — Manual. <http://php.net/mysqli>.
- [13] The Apache Software Foundation. <http://svn.apache.org>