VITELS Hands-on Session Setup

Attila Weyland*

weyland@iam.unibe.ch

University of Bern
Institute of Computer Science and Applied Mathematics
Computer Networks and Distributed Systems
Neubrückstr. 10, CH-3012 Bern

September 2, 2004

Contents

1 Introduction **2**
1.1 Conventions . . . 2

2 Portal Machines **3**
2.1 Installation . . . 3
2.2 Module Specific Installation . . . 5
2.2.1 Simulation of IP Network Configuration . . . 5
2.2.2 Sockets and RPC . . . 5
2.2.3 Remote Method Invocation . . . 6
2.2.4 Application Server . . . 7
2.3 Configuration . . . 7
2.4 Module Specific Configuration . . . 12
2.4.1 Simulation of IP Network Configuration . . . 12
2.4.2 Sockets and RPC . . . 14
2.4.3 Remote Method Invocation . . . 18
2.4.4 Application Server . . . 22

3 Lab Machines **25**
3.1 Installation . . . 25
3.2 Module Specific Installation . . . 26
3.2.1 Simulation of IP Network Configuration . . . 26
3.2.2 Sockets and RPC . . . 27

*with contributions from module authors Christine Rosenberger and Günther Stattenberger

3.2.3 Remote Method Invocation 27
3.2.4 Application Server 28
3.3 Configuration 29
3.4 Module Specific Configuration 29
3.4.1 Simulation of IP Network Configuration 29
3.4.2 Sockets and RPC 31
3.4.3 Remote Method Invocation 32
3.4.4 Application Server 34

4 Virtual Sessions **35**
4.1 Preparation & Deployment 35
4.1.1 Simulation of IP Network Configuration 35
4.1.2 Client/Server Concepts 36

1 Introduction

This document describes the setup of the hands-on part of the modular structured distance learning course named VITELS. The VITELS architecture is shown in Figure 1. The modules can have virtual and/or real hands-on parts. A virtual lab session does not require dedicated hardware and is usually hosted by the course server (e.g. a Java applet). A real lab session typically requires a reservation server, which schedules the access to the lab and a portal server, which controls the access to the lab machine(s). The lab machine hosts the actual exercises.

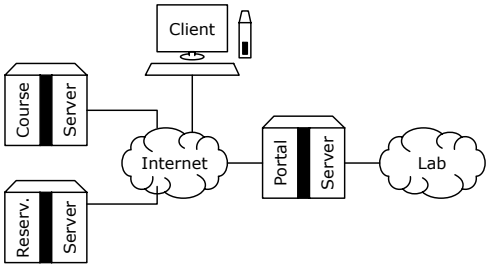

Figure 1: VITELS architecture

Table 1 lists all the VITELS modules and their corresponding number. Currently five modules (2, 4, 8, 9 and 10) are covered by this document.

All machines run Debian GNU/Linux 3.0.

1.1 Conventions

To support fast reading and finding in this document several conventions have been used.

Command lines to be executed in a shell are written in a typewriter face, e.g. `execute me now`. The location of files and directories is written using the package url, e.g. /i/am/a/file or /we/are/directories/. Required or recommended software (debian tasks or

Module	ID	Session Type	
		Virtual	Real
Linux Installation	1	√	—
Simulation of IP Network Configuration	2	√	√[1]
Performance Evaluation	3	—	√[1]
Client/Server Concepts	4	√	—
Protocol Analysis	5	—	√
IP Security	6	—	√[1]
Firewall Management	7	—	√[1]
Sockets and RPC	8	—	√[1]
Remote Method Invocation	9	—	√[1]
Application Server	10	—	√[1]

Table 1: Module names and corresponding numbers

packages or manual installations) can be followed by the application which requires the software enclosed in special braces ⌈⌋, e.g.

- examplary software package ⌈requiring application⌋.

It is assumed that you care for your system and that you know what you are doing.

2 Portal Machines

2.1 Installation

The following Debian tasks (`tasksel`) must be installed:

- web server ⌈lab_portal⌋

The following Debian packages (`dselect`) and their dependencies must be installed:

- libapache-mod-ssl ⌈https⌋
- libapache-mod-ssl-doc ⌈https⌋
- libcurl2-ssl ⌈AAI⌋
- ntp (time.unibe.ch ntp.metas.ch swisstime.ethz.ch)
- ntp-simple
- ntpdate
- php4 ⌈lab_portal⌋

[1] limited simultaneous user access, requires resource management via reservation system

- php4-ldap ⌈AAI⌋
- stunnel ⌈ldaps⌋
- unzip ⌈lab_portal⌋

The following Debian tasks (`tasksel`) may be installed according to the need:

- C and C++

The following Debian packages (`dselect`) and their dependencies may be installed according to the need:

- emacs
- kernel-source-2.2.xx
- mc

The following software must be installed manually:

- lab_portal [7]
- shibboleth target [4]

Lab_portal *root*

1. extract portal/ from the lab_portal archive to /var/www/

Shibboleth Target *root* This paragraph describes how resources can participate in the SWITCHaai [5]. It is recommended that you also read the Shibboleth 1.1 Target Deployment Guide [2]. The following packages and files need to be downloaded from [4]:

- AAP.xml
- apache_1.3.26-0woody3_i386.deb (not required anymore)
- ca-bundle.crt
- httpd.conf.addon
- init.d-apache-debian
- shar.logger
- shib-target-1.1-debian-3.0r1.tar.gz
- shibboleth.ini
- shibboleth.logger
- shire.logger

1. run `tar xvzCf / shib-target-1.1-debian-3.0r1.tar.gz`
2. place or replace the following previously downloaded files in /opt/shibboleth/etc/shibboleth/
   - AAP.xml
   - shar.logger
   - shibboleth.ini
   - shibboleth.logger
   - shire.logger
3. replace /etc/apache/ssl.crt/ca-bundle.crt with the previously downloaded ca-bundle.crt
4. replace /etc/init.d/apache with the previously downloaded init.d-apache-debian

2.2 Module Specific Installation

2.2.1 Simulation of IP Network Configuration

Table 2 contains the settings required during the Debian installation procedure of the portal machine for module 2.

Hostname	coyote
Interface	eth0
IP address	130.92.66.147
Netmask	255.255.255.0
Broadcast	130.92.66.255
Gateway	130.92.66.1
Module	eepro100

Table 2: Module 2 portal installation settings

2.2.2 Sockets and RPC

Table 3 contains the settings required during the Debian installation procedure of the portal machine for module 8.

The following Debian packages (`dselect`) and their dependencies must be installed:

- sudo ⌈socketsrpc-exercise⌋

The following software must be installed manually:

- mindterm ⌈socketsrpc-exercise⌋

Hostname	coyote	
Interface	eth1	eth0
IP address	130.92.66.147	10.1.1.1
Netmask	255.255.255.0	255.255.255.0
Broadcast	130.92.66.255	10.1.1.255
Gateway	130.92.66.1	
Module	eepro100	eepro100

Table 3: Module 8 portal installation settings

Mindterm *root*

1. extract mindterm archive to /var/www/portal/mindterm
2. create a self-signed certificate (the mindterm Java applet needs to be signed so that "Capture To File..." is working)
   a) run `keytool -genkey -keystore myKeystore -alias myself`
   b) run `keytool -selfcert -keystore myKeystore -alias myself`
3. sign the Mindterm jar archive by running `jarsigner -keystore myKeystore /var/www/portal/mindterm/mindterm.jar myself`

2.2.3 Remote Method Invocation

Table 4 contains the settings required during the Debian installation procedure of the portal machine for module 9.

Hostname	vitels7	
Interface	eth0	eth1
IP address	130.92.65.179	10.1.1.1
Netmask	255.255.255.0	255.255.255.0
Broadcast	130.92.65.255	10.1.1.255
Gateway	130.92.65.1	
Module	sis900 (2.4.19)	8139too (2.4.19)

Table 4: Module 9 portal installation settings

The following Debian packages (`dselect`) and their dependencies must be installed:

- sudo

The following software must be installed manually:

- mindterm

Mindterm *root* see corresponding paragraph in Section 2.2.2 on page 6.

2.2.4 Application Server

Table 5 contains the settings required during the Debian installation procedure of the portal machine for module 10. The following Debian packages (`dselect`) and their

Hostname	vitels7	
Interface	eth0	eth1
IP address	130.92.65.179	10.1.1.1
Netmask	255.255.255.0	255.255.255.0
Broadcast	130.92.65.255	10.1.1.255
Gateway	130.92.65.1	
Module	sis900 (2.4.19)	8139too (2.4.19)

Table 5: Module 10 portal installation settings

dependencies must be installed:

- sudo

The following software must be installed manually:

- mindterm

Mindterm *root* see corresponding paragraph in Section 2.2.2 on page 6.

2.3 Configuration

Mod_ssl *root*

1. change into the following directory /etc/apache/
2. run `openssl genrsa -out ssl.key/yourmachine.key 2048`
3. run `openssl req -new -key ssl.key/yourmachine.key -out ssl.csr/yourmachine.csr`
4. run `openssl req -x509 -days 10000 -key ssl.key/yourmachine.key -in ssl.csr/yourmachine.csr -out ssl.crt/yourmachine.crt`
5. enable SSL support for apache in /etc/apache/httpd.conf by uncommenting the following line

   ```
   LoadModule ssl_module /usr/lib/apache/1.3/mod_ssl.so
   ```

6. insert the content of /usr/share/doc/libapache-mod-ssl-doc/mod-ssl.conf into /etc/apache/httpd.conf before the section "Virtual Hosts"
7. enable HTTPS requests support for apache in /etc/apache/httpd.conf by uncommenting the following lines

```
<IfModule mod_ssl.c>
Listen 80
Listen 443
</IfModule>
```

8. add a virtual host for processing HTTPS requests in /etc/apache/httpd.conf by adding the following lines

```
<VirtualHost _default_:80>
Redirect / https://yourmachine/
</VirtualHost>

<VirtualHost _default_:443>
<IfModule mod_ssl.c>
SSLEngine on
SSLCertificateFile /etc/apache/ssl.crt/yourmachine.crt
SSLCertificateKeyFile /etc/apache/ssl.key/yourmachine.key
SSLCACertificateFile /etc/apache/ssl.crt/ca-bundle.crt
SetEnvIf User-Agent ".*MSIE.*" nokeepalive [LF]
  ssl-unclean-shutdown
</IfModule>
</VirtualHost>
```

PHP4 *root*

1. enable PHP4 support for apache in /etc/apache/httpd.conf by uncommenting the following line

```
LoadModule php4_module /usr/lib/apache/1.3/libphp4.so
```

Stunnel *root*

1. create /usr/local/sbin/stunnel_ldap.sh with the following content

```
#!/bin/bash
#
# SSL tunnel for LDAP queries to LDAPS server
#
# -c       enable client mode (remote service uses SSL)
# -d 636   listen for connection on localhost:636
# -r ldap.cnds.unibe.ch:636
#          connect to remote service on ldap.cnds.unibe.ch:636
#
/usr/sbin/stunnel -c -d 636 -r ldap.cnds.unibe.ch:636
```

(there is no need to generate keys, since stunnel is only operating in client mode)
(check the path names)

2. run `chmod 700 /usr/local/sbin/stunnel_ldap.sh`

Lab_portal *root*

1. create directory /var/www/portal/data and restrict access it
   a) run `chmod 0700 data`
   b) run `chown www-data:www-data data`
2. restrict access to LDAP dn and password from the lab portal
   a) run `chmod 0400 portal_settings.inc.php`
   b) run `chown www-data:www-data portal_settings.inc.php`
3. redirect generic request to the lab portal directory by creating /var/www/index.html with the following content

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" [LF]
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title>Lab Portal Redirect</title>
<meta name="Author" content="author's name"/>
<meta http-equiv="Content-Type" [LF]
  content="text/html; charset=iso-8859-1"/>
<meta http-equiv="refresh" content="0; [LF]
  URL=https://yourportalmachine/portal/"/>
</head>
<body>
</body>
</html>
```

4. when linking to the portal page (e.g. from within the course), refer directly to your module's customized portal page (to avoid displaying the module selection list) by using the following URL https://yourportalmachine/portal/entry.php?mod_id=yourmoduleid

Shibboleth Target *root* These steps allow resources to participate in the SWITCHaai (see also Shibboleth Target-Deployment Guide 1.1 at http://www.switch.ch/aai/).

1. obtain a signed certificate from SWITCH for SSL key
   (uses key and certificate request and obsoletes self-signed certificate previously created during Mod_ssl configuration, see Section 2.3 on page 7)

a) send /etc/apache/ssl.crt/yourportalmachine.csr to aai@switch.ch

b) replace /etc/apache/ssl.crt/yourportalmachine.crt with newly obtained certificate signed by SWITCH

2. configure shibboleth to match your machine's ssl key and certificate in /opt/shibboleth/etc/shibboleth/shibboleth.ini by adjusting the following parameters

```
# Should provide a key-pair and certificate
# Can use mod_ssl's server.crt/server.key if you set file
# permissions
certfile=/etc/apache/ssl.crt/yourportalmachine.crt
keyfile=/etc/apache/ssl.key/yourportalmachine.key
#keypass=
calist=/etc/apache/ssl.crt/ca-bundle.crt
```

3. enable shibboleth support for apache by adding the following lines at the end of /etc/apache/httpd.conf

```
######
## SHIB Config
######

LoadModule shibrm_module /opt/shibboleth/libexec/mod_shibrm.so
LoadModule shire_module /opt/shibboleth/libexec/mod_shire.so

# Global SHIRE Configuration
SHIREConfig /opt/shibboleth/etc/shibboleth/shibboleth.ini

# Set the SHIRE POST processor URL
SHIREURL /shibboleth/SHIRE
<Location /shibboleth/SHIRE>
  SetHandler shib-shire-post
</Location>

# authorization restriction for lab portal www directory
<Location /portal>
  AuthType shibboleth
  ShibExportAssertion On
  require valid-user
</Location>
```

4. create directory for shar and shire log files by running `mkdir /var/log/shibboleth`

5. obtain the certificate file from the WAYF server

a) run `openssl s_client -showcerts -connect wayf1.switch.ch:443`

b) copy the first block from the returned output including `BEGIN CERTIFICATE` and `END CERTIFICATE` into /opt/shibboleth/etc/shibboleth/wayf1.switch.ch.crt

6. keep

   - https://wayf1.switch.ch/SWITCHaai/sites.xml
   - https://wayf1.switch.ch/SWITCHaai/trust.xml

   up-to-date

   a) create /usr/local/sbin/refresh_shibboleth.sh with the following content

```
#!/bin/bash
#
# refresh the sites.xml and trust.xml for shibboleth
# walkaround via wget since siterefresh does not support
# proxies.
#
FILE[1]=sites.xml
FILE[2]=trust.xml
http_proxy=proxy.unibe.ch:80

export http_proxy

echo
echo " REFRESH SHIBBOLETH"
echo " ------------------"
echo

cd /opt/shibboleth/etc/shibboleth

for index in 1 2
do
  echo " * verify ${FILE[index]}"
# download the files
  wget -q -t 1 -O ./${FILE[index]}.input \
  http://wayf1.switch.ch/SWITCHaai/${FILE[index]}
  ../../bin/siterefresh --cert wayf1.switch.ch.crt --schema . \
  --url file:./${FILE[index]}.input --out ${FILE[index]}.output

# compare downloaded with existing files and overwrite if
# downloaded files are newer
  cmp ${FILE[index]}.output ${FILE[index]}
  if [ $? -eq 0 ]
```

```
  then
    echo " * no refresh needed for ${FILE[index]}"
  else
    echo " * ${FILE[index]} refreshed"
    cp ${FILE[index]}.output ${FILE[index]}
  fi
done

echo

exit 0
```

(check the path names)

b) run `chmod 700 /usr/local/sbin/refresh_shibboleth.sh`

Set up Cron Jobs *root*

1. set up cron jobs in /etc/crontab by adding the following lines

```
*  3  *  *  7   root   /usr/local/sbin/refresh_shibboleth.sh
@reboot         root   /usr/local/sbin/stunnel_ldap.sh
```

2.4 Module Specific Configuration

2.4.1 Simulation of IP Network Configuration

Create User Accounts *root*

1. run `adduser module2`

Prepare User Data *module2*

1. create the following directory structure

```
/home/module2/wivrec-data/
/home/module2/wivrec-data/scripts/
/home/module2/wivrec-data/topologies/
/home/module2/wivrec-data/users/
```

2. run `chmod 0775 /home/module2/wivrec-data/*`

3. run `chown :portal scripts topologies`

4. run `chown :www-data users`

Export User Data *root*

1. export the user data by adding the following line to /etc/exports

```
/home/module2/wivrec-data 130.92.65.174(rw) 130.92.65.175(rw)
```

Prepare Automated Clean-up *root, www-data*

1. login as *www-data*
2. create /var/www/portal/data/wrapper_reset_simulation_wivrec.sh with the following content

```
#!/bin/sh
#
# wrapper script calls the renew/reset script and enables to
# block on it until the it has finished
{
nohup sudo /root/reset_simulation_wivrec.sh $1
} >/dev/null 2>&1 &
```

3. create /root/reset_simulation_wivrec.sh with the following content

```
#!/bin/bash
#
# reset script for module 2 (wivrec)

WIVREC=/home/module2/wivrec-data

if [[ "x$1" == x ]]
then
    exit 1
fi
# delete wivrec account data for the specified user
rm -rf ${WIVREC}/scripts/$1/
rm -rf ${WIVREC}/topologies/$1.bin
```

Restrict Access *root*

1. disable access for everyone by adding the following lines to /etc/host.deny

```
portmap:ALL
mountd:ALL
rquotad:ALL
statd:ALL
```

2. enable access for VR machines by adding the following lines to /etc/host.allow to /etc/host.deny

```
portmap: 130.92.65.174, 130.92.65.175
lockd: 130.92.65.174, 130.92.65.175
rquotad: 130.92.65.174, 130.92.65.175
mountd: 130.92.65.174, 130.92.65.175
statd: 130.92.65.174, 130.92.65.175
```

2.4.2 Sockets and RPC

Add Host Names *root*

1. add the following lines to /etc/hosts

```
10.1.1.10       challenger
10.1.1.11       enterprise
```

Create User Accounts *root*

1. create user accounts for user *sockrpc1* and *sockrpc2* (corresponding to the number of lab machines)
   - run `adduser username`

Prepare Automated Login *root*, *sockrpc1*, *sockrpc2*

1. login as *root*
2. generate an authentication key for SSH of type RSA
   - run `ssh-keygen -t rsa` without entering a passphrase
3. copy the public key file to the authorized keys file of the user *root* on challenger and enterprise
   - change into the directory /root/.ssh/
   - run `scp id_rsa.pub challenger:/root/.ssh/authorized_keys`
   - run `scp id_rsa.pub enterprise:/root/.ssh/authorized_keys`
4. login as *sockrpc1*
5. generate an authentication key for SSH of type RSA
   - run `ssh-keygen -t rsa` without entering a passphrase
6. copy the public key file to the authorized keys file of the user *sockrpc1* on challenger

- change into the directory /home/sockrpc1/.ssh/
- run `scp id_rsa.pub challenger:/home/sockrpc1/.ssh/authorized_keys`

7. login as *sockrpc2*
8. generate an authentication key for SSH of type RSA
   - run `ssh-keygen -t rsa` without entering a passphrase
9. copy the public key file to the authorized keys file of the user *sockrpc2* on enterprise
   - change into the directory /home/sockrpc2/.ssh/
   - run `scp id_rsa.pub enterprise:/home/sockrpc2/.ssh/authorized_keys`

Redirect User Shell *root*

1. redirect the shell for each created lab user account
   - change the following lines in /etc/passwd

     ```
     sockrpc1:x:1000:1000:,,,:/home/sockrpc1:/bin/bash
     sockrpc2:x:1000:1000:,,,:/home/sockrpc2:/bin/bash
     ```

     to

     ```
     sockrpc1:x:1000:1000:,,,:/home/sockrpc1:/root/sockrpc1
     sockrpc2:x:1000:1000:,,,:/home/sockrpc2:/root/sockrpc2
     ```

2. create /root/sockrpc1 with the following content

   ```
   #!/bin/sh
   #
   # open a SSH connection to a lab machine
   #
   /usr/bin/ssh challenger
   ```

   (this detour via the shell script is required since /etc/passwd does not take spaces in the shell option)

3. run `chmod 755 /root/sockrpc1`
4. create /root/sockrpc2 with the following content

   ```
   #!/bin/sh
   #
   # open a SSH connection to a lab machine
   #
   /usr/bin/ssh enterprise
   ```

(this detour via the shell script is required since /etc/passwd does not take spaces in the shell option)

5. run `chmod 755 /root/sockrpc2`

Prepare Automated Clean-up and Password Generation *root, www-data*

1. login as *www-data*
2. create /var/www/portal/data/wrapper_renew_sockets_hosts.sh with the following content

```
#!/bin/sh
#
# wrapper script calls the renew/reset script and enables to
# block on it until it has finished
{
nohup sudo /root/renew_sockets_hosts.sh
} >/dev/null 2>&1 &
```

3. create /var/www/portal/data/wrapper_reset_sockets_challenger.sh with the following content

```
#!/bin/sh
#
# wrapper script calls the renew/reset script and enables to
# block on it until it has finished
{
nohup sudo /root/reset_sockets_challenger.sh
} >/dev/null 2>&1 &
```

4. create /var/www/portal/data/wrapper_reset_sockets_enterprise.sh with the following content

```
#!/bin/sh
#
# wrapper script calls the renew/reset script and enables to
# block on it until it has finished
{
nohup sudo /root/reset_sockets_enterprise.sh
} >/dev/null 2>&1 &
```

5. login as *root*
6. create /root/renew_sockets_hosts.sh with the following content

```
#!/bin/bash
#
# renew script for module 8

# delete old and create new user directories
ssh root@challenger "rm -rf /home/sockrpc1;cp -a [LF]
  /root/sockrpc1/ /home"
ssh root@enterprise "rm -rf /home/sockrpc2;cp -a [LF]
  /root/sockrpc2/ /home"

# create new password
# generate 6 pseudo random bytes of 8 bit length and apply
# Base64 encoding to them resulting in 8 characters of 6 bit
# length
passwd=‘openssl rand -base64 6‘

# change user passwords
(echo $passwd;sleep 1;echo $passwd)|passwd sockrpc1 [LF]
  2>/dev/null
(echo $passwd;sleep 1;echo $passwd)|passwd sockrpc2 [LF]
  2>/dev/null

# change applet parameters
sed /password/s!value=\".......\"!value=\"$passwd\"! [LF]
  /var/www/portal/module_8/module_8.php > temp

cp temp /var/www/portal/module_8/module_8.php

rm -f temp
```

(/root/renew_sockets_hosts.sh is going to be called from /var/www/portal/module_8/module_8.php via /var/www/portal/data/wrapper_renew_sockets_hosts.sh during login)

7. create /root/reset_sockets_challenger.sh with the following content

```
#!/bin/bash
#
# reset script for module 8 (challenger)

# delete old user directory and create new user directory
ssh root@challenger "rm -rf /home/sockrpc1;cp -a /root/sockrpc1/ /home"
```

8. create /root/reset_sockets_enterprise.sh with the following content

```
#!/bin/bash
#
# reset script for module 8 (enterprise)

# delete old user directory and create new user directory
ssh root@enterprise "rm -rf /home/sockrpc2;cp -a /root/sockrpc2/ /home"
```

9. allow user *www-data* to execute the renew/reset scripts as user *root*

   - add the following line to /etc/sudoers

     ```
     www-data localhost=NOPASSWD:/root/renew_sockets_hosts.sh
     www-data coyote=NOPASSWD:/root/renew_sockets_hosts.sh
     www-data localhost=NOPASSWD:/root/reset_sockets_challenger.sh
     www-data coyote=NOPASSWD:/root/reset_sockets_challenger.sh
     www-data localhost=NOPASSWD:/root/reset_sockets_enterprise.sh
     www-data coyote=NOPASSWD:/root/reset_sockets_enterprise.sh
     ```

Populate Known Hosts List *root*

1. run `ssh sockrpc1@coyote` and type `y` to add this host to the known hosts list
2. run `ssh sockrpc2@coyote` and type `y` to add this host to the known hosts list
3. run `ssh root@challenger` and type `y` to add this host to the known hosts list
4. run `ssh root@enterprise` and type `y` to add this host to the known hosts list

Disable IP Forwarding *root*

1. run `cat 0>/proc/sys/net/ipv4/ip_forward`

2.4.3 Remote Method Invocation

Add Host Names *root*

1. add the following lines to /etc/hosts

   ```
   10.1.1.20       martian
   10.1.1.21       elmer
   ```

Create User Accounts *root*

1. create user accounts for user *rmirpc1* and *rmi2* (corresponding to the number of lab machines)

   - run `adduser username`

Prepare Automated Login *root, rmi1, rmi2*

1. login as *root*
2. generate an authentication key for SSH of type RSA
   - run `ssh-keygen -t rsa` without entering a passphrase
3. copy the public key file to the authorized keys file of the user *root* on martian and elmer
   - change into the directory /root/.ssh/
   - run `scp id_rsa.pub martian:/root/.ssh/authorized_keys`
   - run `scp id_rsa.pub elmer:/root/.ssh/authorized_keys`
4. login as *rmi1*
5. generate an authentication key for SSH of type RSA
   - run `ssh-keygen -t rsa` without entering a passphrase
6. copy the public key file to the authorized keys file of the user *rmi1* on martian
   - change into the directory /home/rmi1/.ssh/
   - run `scp id_rsa.pub martian:/home/rmi1/.ssh/authorized_keys`
7. login as *rmi2*
8. generate an authentication key for SSH of type RSA
   - run `ssh-keygen -t rsa` without entering a passphrase
9. copy the public key file to the authorized keys file of the user *rmi2* on elmer
   - change into the directory /home/rmi2/.ssh/
   - run `scp id_rsa.pub elmer:/home/rmi2/.ssh/authorized_keys`

Redirect User Shell *root*

1. redirect the shell for each created lab user account
   - change the following lines in /etc/passwd

     ```
     rmi1:x:1001:1001:,,,:/home/rmi1:/bin/bash
     rmi2:x:1002:1002:,,,:/home/rmi2:/bin/bash
     ```

     to

     ```
     rmi1:x:1001:1001:,,,:/home/rmi1:/root/rmi1
     rmi2:x:1002:1002:,,,:/home/rmi2:/root/rmi2
     ```

2. create /root/rmi1 with the following content

```
#!/bin/sh
#
# open a SSH connection to a lab machine
#
/usr/bin/ssh martian
```

   (this detour via the shell script is required since /etc/passwd does not take spaces in the shell option)

3. run `chmod 755 /root/rmi1`

4. create /root/rmi2 with the following content

```
#!/bin/sh
#
# open a SSH connection to a lab machine
#
/usr/bin/ssh elmer
```

   (this detour via the shell script is required since /etc/passwd does not take spaces in the shell option)

5. run `chmod 755 /root/rmi2`

Prepare Automated Clean-up and Password Generation *root, www-data*

1. login as *www-data*

2. create /var/www/portal/data/wrapper_renew_rmi_hosts.sh with the following content

```
#!/bin/sh
#
# wrapper script calls the renew script and enables to
# block on it until it has finished
{
nohup sudo /root/renew_rmi_hosts.sh
} >/dev/null 2>&1 &
```

3. create /var/www/portal/data/wrapper_reset_rmi_server.sh with the following content

```
#!/bin/sh
#
# wrapper script calls the reset script and enables to
```

```
# block on it until it has finished
{
nohup sudo /root/reset_rmi_server.sh
} >/dev/null 2>&1 &
```

4. create /var/www/portal/data/wrapper_reset_rmi_client.sh with the following content

```
#!/bin/sh
#
# wrapper script calls the reset script and enables to
# block on it until it has finished
{
nohup sudo /root/reset_rmi_client.sh
} >/dev/null 2>&1 &
```

5. login as *root*

6. create /root/reset_rmi_hosts.sh with the following content

```
#!/bin/bash
#
# renew script for module 9

# delete old and create new user directories
ssh root@martian "rm -rf /home/rmi1;cp -a /root/rmi1/ /home"
ssh root@elmer "rm -rf /home/rmi2;cp -a /root/rmi2/ /home"

# create new password
# generate 6 pseudo random bytes of 8 bit length and apply
# Base64 encoding to them resulting in 8 characters of 6 bit
# length
passwd=‘openssl rand -base64 6‘

# change user passwords
(echo $passwd;sleep 1;echo $passwd)|passwd rmi1 2>/dev/null
(echo $passwd;sleep 1;echo $passwd)|passwd rmi2 2>/dev/null

# change applet parameters
sed /password/s!value=\".......\"!value=\"$passwd\"! [LF]
  /var/www/portal/module_9/module_9.php > temp

cp temp /var/www/portal/module_9/module_9.php

rm -f temp
```

(/root/renew_rmi_hosts.sh is going to be called from /var/www/portal/module_9/module_9.php via /var/www/portal/data/wrapper_renew_rmi_hosts.sh during login)

7. create /root/reset_rmi_server.sh with the following content

```
#!/bin/bash
#
# reset script for module 9 (server)

# delete old and create new user directories
ssh root@martian "rm -rf /home/rmi1;cp -a /root/rmi1/ /home"
```

8. create /root/reset_rmi_client.sh with the following content

```
#!/bin/bash
#
# reset script for module 9 (client)

# delete old and create new user directories
ssh root@elmer "rm -rf /home/rmi2;cp -a /root/rmi2/ /home"
```

9. allow user *www-data* to execute /root/renew_rmi_hosts.sh, /root/reset_rmi_server.sh and /root/reset_rmi_client.sh as user *root*

   - add the following line to /etc/sudoers

```
www-data localhost=NOPASSWD:/root/renew_rmi_hosts.sh
www-data vitels7=NOPASSWD:/root/renew_rmi_hosts.sh
www-data localhost=NOPASSWD:/root/reset_rmi_server.sh
www-data vitels7=NOPASSWD:/root/reset_rmi_server.sh
www-data localhost=NOPASSWD:/root/reset_rmi_client.sh
www-data vitels7=NOPASSWD:/root/reset_rmi_client.sh
```

Populate Known Hosts List *root*

1. run `ssh rmi1@vitels7` and type `y` to add this host to the known hosts list
2. run `ssh rmi2@vitels7` and type `y` to add this host to the known hosts list
3. run `ssh root@martian` and type `y` to add this host to the known hosts list
4. run `ssh root@elmer` and type `y` to add this host to the known hosts list

2.4.4 Application Server

Add Host Names *root*

1. add the following lines to /etc/hosts

```
10.1.1.35       kif
```

Create User Accounts *root*

1. create user account for user *apps0* (corresponding to the number of lab machines)
   - run `adduser username`

Prepare Automated Login *root, apps0*

1. login as *root*
2. generate an authentication key for SSH of type RSA
   - run `ssh-keygen -t rsa` without entering a passphrase
3. copy the public key file to the authorized keys file of the user *root* on kif
   - change into the directory /root/.ssh/
   - run `scp id_rsa.pub kif:/root/.ssh/authorized_keys`
4. login as *apps0*
5. generate an authentication key for SSH of type RSA
   - run `ssh-keygen -t rsa` without entering a passphrase
6. copy the public key file to the authorized keys file of the user *apps0* on kif
   - change into the directory /home/apps0/.ssh/
   - run `scp id_rsa.pub kif:/home/apps0/.ssh/authorized_keys`

Redirect User Shell *root*

1. redirect the shell for each created lab user account
   - change the following lines in /etc/passwd

     `apps0:x:1003:1003:,,,:/home/apps0:/bin/bash`

     to

     `apps0:x:1003:1003:,,,:/home/apps0:/root/apps0`
2. create /root/apps0 with the following content

   ```
   #!/bin/sh
   #
   # open a SSH connection to a lab machine
   #
   /usr/bin/ssh kif
   ```

   (this detour via the shell script is required since /etc/passwd does not take spaces in the shell option)
3. run `chmod 755 /root/apps0`

Prepare Automated Clean-up and Password Generation *root, www-data*

1. login as *www-data*
2. create /var/www/portal/data/wrapper_renew_application_host.sh with the following content

```
#!/bin/sh
#
# wrapper script calls the renew script and enables to
# block on it until it has finished
{
nohup sudo /root/renew_application_host.sh
} >/dev/null 2>&1 &
```

3. create /var/www/portal/data/wrapper_reset_application_host.sh with the following content

```
#!/bin/sh
#
# wrapper script calls the reset script and enables to
# block on it until it has finished
{
nohup sudo /root/reset_application_host.sh
} >/dev/null 2>&1 &
```

4. login as *root*
5. create /root/renew_application_host.sh with the following content

```
#!/bin/bash
#
# renew script for module 10
#
# delete old and create new user directories
ssh root@kif "rm -rf /home/apps0;cp -a /root/apps0/ /home"

# create new password
# generate 6 pseudo random bytes of 8 bit length and apply
# Base64 encoding to them resulting in 8 characters of 6 bit
# length
passwd=‘openssl rand -base64 6‘

# change user passwords
(echo $passwd;sleep 1;echo $passwd)|passwd apps0 2>/dev/null
```

```
# change applet parameters
sed /password/s!value=\".......\"!value=\"$passwd\"! [LF]
  /var/www/portal/module_10/module_10.php > temp

cp temp /var/www/portal/module_10/module_10.php

rm -f temp
```

(/root/renew_application_host.sh is going to be called from /var/www/portal/module_10/module_10.php via /var/www/portal/data/wrapper_renew_application_host.sh during login)

6. create /root/reset_application_host.sh with the following content

```
#!/bin/bash
#
# reset script for module 10
#
# delete old and create new user directories
# ensure that no application processes are running
ssh root@kif "rm -rf /home/apps0;cp -a /root/apps0/ /home"
ssh root@kif "exec /root/clean.sh"
```

7. allow user *www-data* to execute /root/renew_application_host.sh and /root/reset_application_host.sh as user *root*

   - add the following line to /etc/sudoers

     ```
     www-data localhost=NOPASSWD:/root/renew_application_host.sh
     www-data vitels7=NOPASSWD:/root/renew_application_host.sh
     www-data localhost=NOPASSWD:/root/reset_application_host.sh
     www-data vitels7=NOPASSWD:/root/reset_application_host.sh
     ```

Populate Known Hosts List *root*

1. run `ssh apps0@vitels7` and type `y` to add this host to the known hosts list
2. run `ssh root@kif` and type `y` to add this host to the known hosts list

3 Lab Machines

3.1 Installation

The following Debian packages (`dselect`) and their dependencies must be installed:

- ntp (time.unibe.ch ntp.metas.ch swisstime.ethz.ch)
- ntp-simple
- ntpdate

The following Debian tasks (`tasksel`) may be installed according to the need:

- C and C++

The following Debian packages (`dselect`) and their dependencies may be installed according to the need:

- emacs
- kernel-source-2.2.xx
- mc

3.2 Module Specific Installation

3.2.1 Simulation of IP Network Configuration

Table 6 contains the settings required during the Debian installation procedure of lab machines for module 2.

Hostname	vitels4	vitels5
Interface	eth0	eth0
IP address	130.92.65.174	130.92.65.175
Netmask	255.255.255.0	255.255.255.0
Broadcast	130.92.65.255	130.92.65.255
Gateway	130.92.65.1	130.92.65.1
Module	sis	sis

Table 6: Module 2 lab installation settings

The following Debian tasks (`tasksel`) must be installed:

- web server ⌈wivrec-web⌋
- C and C++ ⌈microvar⌋

The following Debian packages (`dselect`) and their dependencies must be installed:

- libapache-mod-ssl ⌈https⌋
- libapache-mod-ssl-doc ⌈https⌋
- php4 ⌈wivrec-web⌋
- xutils ⌈microvar⌋

The following software must be installed manually:

- Java 2 SDK 1.2.2 [3] ⌈wivrec-admin, wivrec-web⌋
- microvar
- wivrec-admin
- wivrec-web

Java 2 SDK 1.2.2 *root*

1. extract the Java 2 SDK archive to /usr/local/jdk1.2.2/

3.2.2 Sockets and RPC

Table 7 contains the settings required during the Debian installation procedure of lab machines for module 8.

Hostname	challenger	enterprise
Interface	eth0	eth0
IP address	10.1.1.10	10.1.1.11
Netmask	255.255.255.0	255.255.255.0
Broadcast	10.1.1.255	10.1.1.255
Gateway	10.1.1.1	10.1.1.1
Module	eepro100	3c59x

Table 7: Module 8 lab installation settings

The following Debian tasks (`tasksel`) must be installed:

- C and C++ ⌈socketsrpc-exercise⌋

The following Debian packages (`dselect`) and their dependencies must be installed:

- emacs ⌈socketsrpc-exercise⌋

The following software must be installed manually:

- socketsrpc-exercise

3.2.3 Remote Method Invocation

Table 8 contains the settings required during the Debian installation procedure of lab machines for module 9.

The following Debian tasks (`tasksel`) must be installed:

- C and C++

The following Debian packages (`dselect`) and their dependencies must be installed:

Hostname	martian	elmer
Interface	eth1	eth1
IP address	10.1.1.20	10.1.1.21
Netmask	255.255.255.0	255.255.255.0
Broadcast	10.1.1.255	10.1.1.255
Gateway	10.1.1.1	10.1.1.1
Module	3c59x	eepro100

Table 8: Module 9 lab installation settings

•

The following software must be installed manually:

- Java 2 SDK 1.4.1 [3] ⌈rmi-exercise⌋

Java 2 SDK 1.4.1 *root*

1. download self-extracting archive from [3] to /usr/local/
2. extract by running `j2sdk-1.4.1-XX-linux-sparc-gcc3.2.bin`
3. create a link by running `ln -s j2sdk1.4.1 jdk`

3.2.4 Application Server

Table 9 contains the settings required during the Debian installation procedure of lab machines for module 10.

Hostname	kif
Interface	eth0
IP address	10.1.1.35
Netmask	255.255.255.0
Broadcast	10.1.1.255
Gateway	10.1.1.1
Module	via-rhine

Table 9: Module 10 lab installation settings

The following Debian tasks (`tasksel`) must be installed:

- C and C++

The following Debian packages (`dselect`) and their dependencies must be installed:

- unzip

The following software must be installed manually:

- Ant 1.5.4 [6] ⌈application-exercise⌋
- Java 2 SDK 1.4.1 [3] ⌈JBoss⌋
- JBoss 3.2.1 [1] ⌈application-exercise⌋

Java 2 SDK 1.4.1 *root*

1. download self-extracting archive from [3] to /usr/local/
2. extract by running `j2sdk-1.4.1-XX-linux-i586.bin`
3. create a link by running `ln -s j2sdk1.4.1 jdk`

JBoss 3.2.1 *root*

1. download zip archive from [1] to /usr/local/
2. extract by running `unzip jboss-3.2.1_tomcat-4.1.24.zip`

Ant 1.5.4 *root*

1. download zip archive from [6] to /usr/local/
2. extract by running `unzip apache-ant-1.5.4-bin.zip`

3.3 Configuration

Mod_ssl *root* see corresponding paragraph in Section 2.3 on page 7.

PHP4 *root* see corresponding paragraph in Section 2.3 on page 8.

3.4 Module Specific Configuration

3.4.1 Simulation of IP Network Configuration

Create User Accounts *root*

1. create a user account on each lab machine, which corresponds to the account created on the portal server
   - run `adduser module2`

Java 2 SDK 1.2.2 *root*

1. extend the PATH and CLASSPATH environment variables in /home/module2/.bash_profile with the following entries

```
export PATH=${PATH}:/usr/local/jdk1.2.2/bin
export CLASSPATH=.
```

Microvar *module2*

1. extract the microvar archive to /home/module2/microvar/ and change into that directory
2. remove the following entry from Makefile under the target "install:"

   ```
   install-doc
   ```

3. run `make;make install`

Wivrec-admin *module2*

1. extract the wivrec-admin archive to /home/module2/wivrec-admin/ and change into that directory
2. edit the following variables to match the machine's configuration in src/Makefile

   ```
   HOSTNAME    = localhost
   PORT        = 9101
   VRDIR       = /home/module2/mv
   J2SDKBINDIR = /usr/local/jdk1.2.2/bin
   ```

3. extract the Java 2 SDK archive to /usr/local/jdk1.2.2/ (usually it suffices to change the variable "HOSTNAME" to the appropriate value)
4. run `make config;make all;make install;make script`

Wivrec-web *root*

1. extract the wivrec-web archive to /root/wivrec-web/ and change into that directory
2. edit the following variables to match the machine's configuration in src/Makefile

   ```
   PREFIX_URL = https://localhost/wivrec/
   PORT       = 9101
   ```

   (usually it suffices to change the variable "PREFIX_URL" to the appropriate value)
3. run `make config;make all;make install`
4. copy the content of www/ to /var/www/wivrec/
5. copy lib/wivrec-web.jar to /var/www/wivrec/
6. edit the following variable to match the machine's configuration in the /var/www/wivrec/lab_settings.inc.php

```
$portal_page = "https://coyote.unibe.ch/portal/";
```

7. redirect generic request to the lab portal directory by creating /var/www/index.html with the following content

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" [LF]
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title>Lab Portal Redirect</title>
<meta name="Author" content="author's name"/>
<meta http-equiv="Content-Type" [LF]
  content="text/html; charset=iso-8859-1"/>
<meta http-equiv="refresh" content="0; [LF]
  URL=https://yourlabmachine/wivrec/"/>
</head>
<body>
</body>
</html>
```

Mount User Data *root*

1. empty the directory /home/module2/wivrec-admin/data/
2. add the following line to /etc/fstab

```
130.92.66.147:/home/module2/wivrec-data
  /home/module2/wivrec-admin/data
  nfs  rsize=1024,wsize=1024,hard,intr 0       0
```

(the nfs address needs to point to the appropriate nfs share)

Set up Cron Jobs *root*

1. set up cron jobs in /etc/crontab by adding the following lines

```
*  3  *  *  *   root   /sbin/reboot
@reboot         module2   [LF]
  /home/module2/wivrec-admin/resources/start/wivrec.sh
```

3.4.2 Sockets and RPC

Add Host Names *root*

1. add the following lines to /etc/hosts

```
10.1.1.1        coyote
10.1.1.10       challenger
10.1.1.11       enterprise
```

Create User Account *root*

1. create a user account on each lab machine, which corresponds to one of the accounts created on the portal server
    - run `adduser sockrpc[X]` where `[X]` should be a number

Prepare Automated Login *root, sockrpc[X]*

1. login as *root*
2. generate an authentication key for SSH of type RSA
    - run `ssh-keygen -t rsa` without entering a passphrase
3. append the public key .ssh/id_rsa.pub file to the authorized keys file portal:/root/.ssh/authorized_keys of the user *root* on the portal
4. login as *sockrpc[X]*
5. generate an authentication key for SSH of type RSA
    - run `ssh-keygen -t rsa` without entering a passphrase
6. copy the public key file to the authorized keys file of the user *sockrpc[X]* on the portal
    - change into the directory /home/sockrpc[X]/.ssh/
    - run `scp id_rsa.pub portal:/home/sockrpc[X]/.ssh/authorized_keys`

Prepare User Data *sockrpc[X]*

1. extract rpcexercise2_server from the socketsrpc-exercise archive tcpexercise2_server to /home/sockrpc[X]/

Backup User Data *root*

1. on each lab machine copy /home/sockrpc[X]/ to /root/sockrpc[X]/ where [X] should be a number

3.4.3 Remote Method Invocation

Add Host Names *root*

1. add the following lines to /etc/hosts

```
10.1.1.1        vitels7
10.1.1.20       martian
10.1.1.21       elmer
```

Create User Account *root*

1. create a user account on each lab machine, which corresponds to one of the accounts created on the portal server
   - run `adduser rmi[X]` where `[X]` should be a number

Java 2 SDK 1.4.1 *root*

Martian

1. extend the PATH and CLASSPATH environment variables in /home/rmi2/.bash_profile with the following entries

```
export PATH=${PATH}:/usr/local/jdk/bin:/home/rmi1
export CLASSPATH=.:/home/rmi1/Fazuul/classes
```

Elmer

1. extend the PATH and CLASSPATH environment variables in /home/rmi2/.bash_profile with the following entries

```
export PATH=${PATH}:/usr/local/jdk/bin:/home/rmi2
export CLASSPATH=.:/home/rmi2/Fazuul/classes: [LF]
  /home/rmi2/Mastermind/Template/classes
```

Prepare Automated Login *root, rmi[X]*

1. login as *root*
2. generate an authentication key for SSH of type RSA
   - run `ssh-keygen -t rsa` without entering a passphrase
3. append the public key .ssh/id_rsa.pub file to the authorized keys file portal:/root/.ssh/authorized_keys of the user *root* on the portal
4. login as *rmi[X]*
5. generate an authentication key for SSH of type RSA
   - run `ssh-keygen -t rsa` without entering a passphrase
6. copy the public key file to the authorized keys file of the user *rmi[X]* on the portal
   - change into the directory /home/rmi[X]/.ssh/
   - run `scp id_rsa.pub portal:/home/rmi[X]/.ssh/authorized_keys`

Prepare User Data *rmi[X]*

1. on each lab machine extract the corresponding archive (martian_rmi1_exercise.zip, elmer_rmi2_exercise.zip to /home/rmi[X]/ where [X] should be a number

Backup User Data *root*

1. on each lab machine copy /home/rmi[X]/ to /root/rmi[X]/ where [X] should be a number

Set up Cron Jobs *root*

1. set up cron jobs in /etc/crontab by adding the following lines

```
* 3 * * * root /sbin/reboot
```

3.4.4 Application Server

Add Host Names *root*

1. add the following lines to /etc/hosts

```
10.1.1.1        vitels7
10.1.1.35       kif
```

Create User Accounts *root*

1. create a user account on the lab machine, which corresponds to the account created on the portal server
   - run `adduser apps0`

Java 2 SDK 1.4.1 *root*

1. extend the PATH and CLASSPATH environment variables in /home/apps0/.bash_profile with the following entries

```
export JAVA_HOME=/usr/local/jdk
export JBOSS_HOME=/usr/local/jboss
export ANT_HOME=/usr/local/apache-ant-1.5.4
export JAVA_OPTS=-client
export PATH=${PATH}:${JAVA_HOME}/bin:/home/apps0: [LF]
  ${JBOSS_HOME}/bin:${JAVA_HOME}/bin/jre/bin:${ANT_HOME}/bin
export CLASSPATH=${JBOSS_HOME}/bin/run.jar: [LF]
  ${JAVA_HOME}/lib/tools.jar
```

Prepare Automated Login *root, apps0*

1. login as *root*
2. generate an authentication key for SSH of type RSA
   - run `ssh-keygen -t rsa` without entering a passphrase
3. append the public key .ssh/id_rsa.pub file to the authorized keys file portal:/root/.ssh/authorized_keys of the user *root* on the portal
4. login as *apps0*
5. generate an authentication key for SSH of type RSA
   - run `ssh-keygen -t rsa` without entering a passphrase
6. copy the public key file to the authorized keys file of the user *apps0* on the portal
   - change into the directory /home/apps0/.ssh/
   - run `scp id_rsa.pub portal:/home/apps0/.ssh/authorized_keys`

Prepare User Data *apps0*

1. on the lab machine extract the archive kif_apps0_exercise.zip to /home/apps0/

Backup User Data *root*

1. on the lab machine copy /home/apps0/ to /root/apps0/

Set up Cron Jobs *root*

1. set up cron jobs in /etc/crontab by adding the following lines

```
* 3 * * *  root  /sbin/reboot
```

4 Virtual Sessions

Virtual sessions do not require any dedicated hardware and thus consist of preparing (installation, configuration, compilation) and deploying a software package. The preparation is usually done on your current work station. The final package is then deployed to the course server.

4.1 Preparation & Deployment

4.1.1 Simulation of IP Network Configuration

The following software must be installed manually:

- IProuting (requires Java SDK 1.2.x or above)

IProuting *current user*

1. extract IProuting archive to your home directory /home/currentuser/IProuting
2. change into directory /home/currentuser/IProuting
3. run `build_nix`
4. copy archive web/IProuting-deploy.zip to your course server
5. extract archive IProuting-deploy.zip into your modules' appropriate directory

4.1.2 Client/Server Concepts

The following software must be installed manually:

- HTTPViewNET (requires Java SDK 1.3.x or above)

HTTPViewNET *current user*

1. extract HTTPViewNET archive to your home directory /home/currentuser/HTTPViewNET
2. change into directory /home/currentuser/HTTPViewNET
3. edit the following line in web/HTTPViewNET.jnlp to match the URL location of the directory of your codebase (where you're going to deploy the archive)

   ```
   <jnlp spec="1.0+" [LF]
     codebase="http://location.of.codebase/directory" [LF]
     href="HTTPViewNET.jnlp">
   ```

4. run `build_nix` (you will be asked to enter a password for a selfsigned keystore)
5. copy archive web/HTTPViewNET-deploy.zip to your web server server
6. extract archive HTTPViewNET-deploy.zip into an accessible web directory

References

[1] JBoss Inc. JBoss Application Server. Software available online from http://www.jboss.org/, 2004.

[2] Shibboleth Project. *Shibboleth Target Deployment Guide*, December 2003. Electronic document available online from http://www.switch.ch/aai/docs/shibboleth/internet2/1.1/DEPLOY-GUIDE-TAR%GET-v1.1.html.

[3] Sun Microsystems Inc. Java 2 SDK Standard Edition. Software available online from http://java.sun.com/, 2004.

[4] SWITCH. Shibboleth target installation (sample) files. Software available online from http://www.switch.ch/aai/docs/shibboleth/SWITCH/1.1/target/, 2003.

[5] SWITCH. Authentication and Authorization Infrastructure. Electronic information available online from http://www.switch.ch/aai/, 2004.

[6] The Apache Software Foundation. Apache Ant. Software available online from http://ant.apache.org/, 2004.

[7] Attila Weyland et al. Portal for Remote Network Laboratories. Software available online from http://www.vitels.ch/project/publications.php, May 2004.