WebSOS Installation and Deployment Manual

This manual is contains information on the deployment of a prototype system implementing WebSOS – a mechanism to provide secure and resilient Web services using an overlay network. The overall architecture of WebSOS is modular; most of its components can be replaced with others that provide the same functionality.
For further information on SOS and WebSOS systems please visit:
http://nsl.cs.columbia.edu/projects/sos/

WebSOS Modules:

1. **Captcha human authentication module:**
   This module uses the Graphical Turing Tests to provide a mechanism to differentiate humans from automated bots and zombies
2. **Secure Tunnel Proxylet:**
   This module is a java proxylet which the user downloads after the authentication phase is complete. This module provides the encryption of all subsequent web traffic to be encrypted using SSL with X.509 certificates.
3. **Communication Control Module for the overlay network & the Overlay Network module:**
   These two modules are responsible for creating the Overlay network and distributing the information about the secret servlets. The control module is responsible for the routing of the actual packets through the overlay network and it is written in Java. The actual implementation of the Overlay network is Chord and it is written in C.

Installation Requirements for each module:

1. The captcha human authentication module:
   a. A web server that supports PHP scripts. In our prototype we used the Apache web server with the PHP module from http://www.php.net .
   b. A database to store the captcha images with the ability to communicate with the PHP scripts. In our prototype we used MySQL from http://www.mysql.com , an open source RDBMS but Oracle, and others can be used instead.
2. Secure Tunnel Proxylet:
   a. Java SDK or JRE version 1.3 and above preferably the latest from http://java.sun.com.
   b. A mechanism for the user to obtain the secure proxylet (we used a web server) but any other mechanism is also fine (manual download and load etc).
3. Communication Control Module:
   Java SDK or JRE version 1.3 and above preferably the latest from http://java.sun.com.
4. Overlay Network:
   a. A Unix machine, preferably a linux installation to run our version of Chord P2P network implementation.

The control and overlay modules communicate using various UDP and TCP ports that are configurable from the command line so the machine has to have these ports open both for outgoing and incoming communication.

Installation Instructions:

The files contained in websos.tar.gz are compressed. To decompress them change to the directory you want to have websos and execute:

tar –zxvf websos.tar.gz

Unless otherwise noted we assume that we are in the installation directory of websos.

The distribution after we extract it contains 3 directories:

captcha	contains the implementation of the captcha module
proxylet	contains the implementation of the proxylet module
deploy_sc	contains the implementation of the control and overlay modules

1. The captcha human authentication module:

As we already mentioned the captcha module requires the installation of mysql or another equivalent RDBS.

After installing MYSQL we have to:

a) Create a MYSQL user see http://www.mysql.com/doc/en/Adding_users.html
In our case we just use the user “root” with password “p@ss” (without the quotes)

b) Create the CAPTCHA database with name “captcha”.
For this use the command:

```
root@sos30#  mysqladmin create captcha –p
```

in the prompt you put the password of the user root, in our case “p@ss”

c) Create a table named “Main” in the “captcha” database:

```
root@sos30# mysql captcha  -p < captcha/captcha.sql
```

This command will also populate the table with the data required for the operation of the database.

The next step is to examine the configuration file:

captcha/captcha-web/dbconnect.inc.php4

This file contains all the paths to various directories we need to create:

```
//Mysql Server Location, usually our localhost
$mysql_host="localhost";
//Mysql Database name
$mysql_db="captcha";
// Mysql Table name
$mysql_table="Main";
//Mysql User with read access to the database
$mysql_user="root";
// The password, the file stores the password
// but the user cannot access it directly
$mysql_password="p@ss";

//debug messages ( 0 OFF or 1 ON)
$debug = 0;

// Base Image Directory of the captcha images
$image_directory = "/www-root/captcha/jpgs";

// Base Image Directory, the files are saved
// in $image_tmp_dir/tmp so please make sure that
// this directory is writable by the httpd daemon
// process (usually 777 is fine since the directory is hidden)
$image_tmp_dir = "/www-root/main/images";
//Text for the captcha directory
$answer_directory = "/www-root/captcha";

// Path to Proxylet Include file
$path_to_proxylet = "/www-root/main/proxylet/index_int.html";
```

The previous file assumes the existence of the following directories:

Apache Document Root: /www-root/main (we will call it DOCUMENTROOT from now on)
Images directory: DOCUMENTROOT/images
Images temporary directory: DOCUMENTROOT/images/tmp (with permissions 777)
Captcha directory (in the system NOT INSIDE DOCUMENTROOT in our case /www-root/captcha with the containts of the captcha directory
(cp –R ./captcha directory_for_the_captcha will do the job).

A directory you will keep the captcha-web e.g. DOCUMENTROOT/sos, in this directory we have to copy the data from the installation directory captcha/captcha-web.

Finally the directory DOCUMENTROOT /proxylet needs to have all the data from the installation proxylet directory.

2. The Secure Proxylet Tunnel

As we mentioned before this module is in proxylet directory after we extract the files.
We copy the module to a location that is accessible by the web server e.g:
DOCUMENTROOT /proxylet (as described in the previous section)
This module has to be configured by opening the file:

    proxylet/packages/settings/Settings.java

and locating the section which contains:

```
//Default Settings change appropriately
put(SERVER_HOSTNAME, "sos30.cs.columbia.edu");
put(SERVER_PORT, new Integer(18080));
put(APPLET_PORT, new Integer(8080));
//Change only in case we test the module we connect
  put(PROXY_HOSTNAME, "sos30.cs.columbia.edu");
  put(PROXY_PORT, new Integer(8080));
//SSL Protocol and Key Specifications
  put(SSL_CONTEXT_PROTOCOL, "SSLv3");
  put(KEY_MANAGER_FACTORY_ALGORITHM, "SunX509");
  put(TRUST_MANAGER_FACTORY_ALGORITHM, "SunX509");
  put(KEY_STORE_TYPE, "JKS");
  put(KEY_PASSWORD, "test12345");
  put(KEY_STORE_PASSWORD, "test12345");
   //Key Location both for the server and the remote client
  put(KEY_FILE_DIRECTORY, "/www-root/main/proxylet/packages/settings/keyFile");
  put(KEY_URL, "http://sos30.cs.columbia.edu/proxylet/packages/settings/keyFile");
```

In this file we need to change the lines containing “sos30.cs.columbia.edu” to the actual host we install WebSOS. The PORTS settings can be changed but in case we do so we need to change the ports in the COMMUNICATION module also.
In addition we need to change both the KEY_FILE_DIRECTORY and the KEY_URL to point to the correct location of the KeyFile.
Finally you need to change to DOCUMENTROOT /proxylet/packages and run the “make” command to incorporate the changes to the corresponding binaries.

3 Communication Control Module for the overlay network & the Overlay Network module:

This module does not need to be accessible from the Web Server we can activate it by copying it to a convenient location and use the “sosctl.pl” script.
ere is the code for chord/sos. To run on a machine, run sosctl.pl as
follows:
./sosctl <BootStrap IP ADDR> <BootStrap Port> <targets file|empty file> <Node name e.g sos30.cs.columbia,edu> <Node Port>.

Please be carefull to give the bootstrap IP Address not the name.
For the first node joining the P2P network give its own ip address as
<BootStrap IP ADDR>
Also targets contains the IP Addresses of the hosts we are the secret servlet
where chord_port can be any unused port, bootstrap should be the ip of the
Chord bootstrap node (i.e. the first node up; use itself for the first
one) and targets should be a list of targets for which that node is the

servlet. All else should happen automatically.

Example for the first host:

```
root@sos30:[~/soscode/deploy_sc]>./sosctl.pl 128.59.23.204 10000 notargets
sos30.cs.columbia.edu 10000
```

And for the another host on the P2P network:

```
/sosctl.pl 128.59.23.204 10000 notargets <new host IP ADDR> <PORT>
```

Remember that one of machines should be configured to be the secret servlet:

```
/sosctl.pl 128.59.23.204 10000 targets <new host IP ADDR> <PORT>
                                ^^^^^^
```

Where as we mentioned targets is a file with the IP Addresses of the hosts we protect. The file we provide has the IP addresses for the site web.mit.edu.

If everything goes OK you should be able to see in the file output.chord something like:

```
root@sos30:[~/soscode/deploy_sc]>more output.chord
Connect to: 128.59.23.204:10000
18.7.21.70
18.7.22.69
18.7.21.69
212.58.240.111
212.58.240.120
18.7.21.116
woke up
18.7.21.70
received announcement for 128.59.23.204
18.7.22.69
received announcement for 128.59.23.204
18.7.21.69
received announcement for 128.59.23.204
212.58.240.111
received announcement for 128.59.23.204
212.58.240.120
received announcement for 128.59.23.204
18.7.21.116
received announcement for 128.59.23.204
```

To stop the Overlay Network we issue:

```
./sosctrl stop
```

TroubleShooting:

In case you receive BAD CERTIFICATE the problem is probably that the CERTIFICATE is EXPIRED.

This is instructions on creating SunX509 Certificates used between the Java Applet and the proxylet Server. The pair is created in the machine that hosts the Applet and the Captcha Web Page and maybe different that the proxylet machine.

To create a X509 Certificate we use the command "keytool" that is part of J2SDK/J2RE (for versions 1.2 and later). For general information on this command look at
http://java.sun.com/products/jdk/1.2/docs/tooldocs/solaris/keytool.html

We execute the following commands:

```
>keytool -genkey -alias main -keypass test12345
```

Here test12345 is the password defined in settings/Settings.java file and can be changed In the "Enter Keystore password" we type once more "test12345" (is the same for the certificate)

We answer to the next questions of the commands in order to create the different fields of the certificate Then in our home directory there is the file ~/.keystore that is the KeyFile. We copy this file to the proxylet/packages/settings (defined again in Settings.java)

```
>cp ~/.keystore (...)proxylet/packages/settings/keyFile
```

Finally we export from the key file the X509 Certificate:

```
>keytool -export -alias main -rfc -file
(...)proxylet/packages/settings/serverKey.cer
```

THIS CERTIFICATE WILL EXPIRE IN 6 Months from the day of creation and we have to create another one.