s2dr

Secure Shared Data Repository for CS6238 Secure Computer Systems

Prerequisites

- Node.js (v4 is required)
- Npm (v3 is required, comes usually with node.js)
- OpenSSL (v0.9.8zg, preinstalled with OSX/Linux)

Tested on:

- OS X El Capitan 10.11.1
- node 4.2.0
- npm 3.3.6
- openssl 0.9.8zg 14 July 2015

Installing

```
git clone git@github.com:tajo/s2dr.git
cd s2dr
npm install
```

Running

You can run one instance of the server and multiple clients. Also, you can run unit tests. Be in the project root. Notice, that on some systems installed `node` can be aliased to `nodejs`. The project is 100% JavaScript (except some external libs like openssl).

```
node server
node client
node test
```

How to use the client app

First you will be asked to enter your `USERNAME`. If the `USERNAME` was never used, it creates a new workspace, generates new keys and signs the public key by CA. If the `USERNAME` was already used (the `workspaces/USERNAME` exists), it just tells the program that you want to use this particular workspace.

Then you have to call `init-session HOSTNAME` to start a secure channel. If you leave `HOSTNAME` empty, `https://localhost:4433` will be used as default.

Welcome interface:

```
node client
What is your username (workspace)? USERNAME
s2dr:USERNAME> init-session https://localhost:4433
Welcome back USERNAME! Secure channel is ready!
s2dr:USERNAME> help
```

You can also use normal bash commands (cwd is set to the selected USERNAME workspace):

```
s2dr:USERNAME> ls
s2dr:USERNAME> touch t1.txt
```

List all s2dr available commands, aliases and parameters via:

```
s2dr:USERNAME> help
```

Deliverables / modules testing

Start the server:

```
$ node server
Server is running at https://localhost:4433 ...
```

1. Initialization of the CA using the server

```
$ node client
What is your username (workspace)? client_0
Workspace client_0 created and selected.
s2dr:client_0> init-session
```

The workspace is availabe at `/workspaces/client_0`.

The newly generated keys are at `/workspaces/client_0/.ssl`.

2. Checking-in with INTEGRITY SecurityFlag.

```
1. Initialize a session with the server as the first client
(let's say, client_0) and check in a document "0.txt" with
INTEGRITY SecurityFlag. (For simplicity, you could use the
name of the document as the Document UID). The name of the
document should be provided as argument of your check_in()
implementation.
```

```
$ node client
What is your username (workspace)? client_0
Workspace client_0 created and selected.
s2dr:client_0> touch 0.txt
s2dr:client_0> init-session
Welcome client_0 for the first time! Secure channel is ready!
s2dr:client_0> check-in 0.txt INTEGRITY
Document 0.txt was successfully uploaded!
```

```
2. Show where the checked-in document and its signature will
be located at the server side. Provide a script for verifying
the signature of the copy.
```

The file is located at `/server/documents/HASH_0.txt`.

The signature is located at `/server/documents/HASH_0.txt.signature`.

The file integrity is always automatically checked during checking-out. Or you can manually compare output from openssl library:

```
openssl dgst -sha256 server/documents/HASH_0.txt
openssl rsautl -verify -inkey server/ssl/server-key.pem
-keyform PEM -in server/documents/HASH_0.txt.signature
```

3. Checking-out a document as its owner.

```
1. Using the same session, check out the document you just
uploaded to the server and store it as “0_copy.txt”.
```

```
s2dr:client_0> check-out 0.txt 0_copy.txt
Document 0.txt was saved to /s2dr/workspaces/client_0/0_copy.txt
s2dr:client_0> ls
0_copy.txt
0.txt
```

4. Checking-out a document as a user without the access permission.

```
1. Initialize a session with the server as the second client
(client_1) and check out the document “0.txt” and store it
as “0_copy.txt”.
```

```
$ node client
What is your username (workspace)? client_1
Workspace client_1 created and selected.
s2dr:client_1> init-session
Welcome client_1 for the first time! Secure channel is ready!
s2dr:client_1> check-out 0.txt 0_copy.txt
You don't have checking-out permission.
```

5. Safe-deletion.

```
1. Using the first session, safely delete “0.txt” then attempt
to check it out again and store as “0_copy2.txt”.
```

```
s2dr:client_0> safe-delete 0.txt
The document 0.txt was safely deleted!
s2dr:client_0> check-out 0.txt 0_copy2.txt
You don't have checking-out permission.
```

6. Checking-in and checking-out with CONFIDENTIALITY SecurityFlag.

```
1. Using the first session, check in a second document “1.txt”
with CONFIDENTIALITY SecurityFlag. Show where the server copy
of “1.txt” is located. The copy needs to be encrypted with a
random AES key that is encrypted using server’s public key.
```

```
s2dr:client_0> touch 1.txt
s2dr:client_0> check-in 1.txt CONFIDENTIALITY
Document 1.txt was successfully uploaded!
```

The file is located at `/server/documents/HASH_1.txt.aes`.

The file key is located at `/server/persist/documents`.

```
2. Using the same session, check out the second document you
just uploaded to the server and store it as “1_copy.txt”.
```

```
s2dr:client_0> check-out 1.txt 1_copy.txt
Document 1.txt was saved to /s2dr/workspaces/client_0/1_copy.txt
s2dr:client_0> ls
0_copy.txt
0.txt
1_copy.txt
1.txt
```

```
3. Terminate the first session.
```

```
s2dr:client_0> terminate-session
Have a great day!
```

7. Updating a document.

```
1. Restart the first session, check in the second document “1.txt”
with CONFIDENTIALITY|INTEGRITY SecurityFlag. Show where the
server copy of “1.txt” and its signature are located. The copy
needs to be encrypted with a different AES key that is encrypted
using server’s public key. Use the same script to verify the
signature of the encrypted copy.
```

```
$ node client
What is your username (workspace)? client_0
Workspace client_0 selected.
s2dr:client_0> init-session
Welcome back client_0! Secure channel is ready!
s2dr:client_0> check-in 1.txt BOTH
Document 1.txt was successfully updated!
```

The file is located at `/server/documents/HASH_1.txt.aes`.

The signature is located at `/server/documents/HASH_1.txt.signature`.

The file integrity is always automatically checked during checking-out. Or you can manually compare output from openssl library:

```
openssl dgst -sha256 server/documents/HASH_1.txt.aes
openssl rsautl -verify -inkey server/ssl/server-key.pem
-keyform PEM -in server/documents/HASH_1.txt.signature
```

```
2. Check out “1.txt” and store it as “1_copy2.txt”.
```

```
s2dr:client_0> check-out 1.txt 1_copy2.txt
Document 1.txt was saved to /s2dr/workspaces/client_0/1_copy2.txt
```

8. Checking-in & checking-out delegation without propagation.

```
1. Using the first session, delegate(“1.txt”, “client_1”, 30,
checking-in | checking-out, False). The delegation timeout
should be 30 seconds.
```

```
s2dr:client_0> delegate 1.txt client_1 30 both false
Document 1.txt was delegated to user client_1 with permission
both until TIMESTAMP+30 and propagationFlag set to false!
```

```
2. Using the second session (client_1), check out “1.txt” and
store it as “1_copy.txt”.
```

```
s2dr:client_1> check-out 1.txt 1_copy.txt
Document 1.txt was saved to /s2dr/workspaces/client_1/1_copy.txt
```

```
3. Using the second session, check in a different text
file as “1.txt”.
```

```
s2dr:client_1> touch 1.txt
s2dr:client_1> check-in 1.txt NONE
Document 1.txt was successfully updated!
```

```
4. Using the second session, delegate(“1.txt”, “client_2”, 30,
checking-in | checking-out, False). The delegation timeout
should be 30 seconds.
```

```
s2dr:client_1> delegate 1.txt client_2 30 both false
User client_2 doesn't exist.
```

```
5. Initialize a session with the server as the third client
(client_2), check out the document “1.txt” and store
it as “1_copy.txt”.
```

```
node client
What is your username (workspace)? client_2
Workspace client_2 selected.
s2dr:client_2> init-session
Welcome back client_2! Secure channel is ready!
s2dr:client_2> check-out 1.txt 1_copy.txt
You don't have checking-out permission.
```

```
6. After the delegation expires, using the second session
(client_1), check out “1.txt” and store it as “1_copy2.txt”.
```

```
s2dr:client_1> check-out 1.txt 1_copy2.txt
You don't have checking-out permission.
```

```
7. Using the first session, check out “1.txt” and store it
as “1_copy3.txt”.
```

```
s2dr:client_0> check-out 1.txt 1_copy3.txt
Document 1.txt was saved to /s2dr/workspaces/client_0/1_copy3.txt
```

9. Checking-out delegation with propagation.

```
1. Using the first session, delegate(“1.txt”, “client_1”,
```

```
30, checking-out, True). The delegation timeout should be
30 seconds.
```

```
s2dr:client_0> delegate 1.txt client_1 30 checking-out true
Document 1.txt was delegated to user client_1 with permission
checking-out until TIMESTAMP+30 and propagationFlag set to true!
```

```
2. Using the second session, delegate(“1.txt”, “client_2”,
60, checking-out, False). The delegation timeout should
be 60 seconds.
```

```
s2dr:client_1> delegate 1.txt client_2 60 checking-out false
Document 1.txt was delegated to user client_2 with permission
checking-out until TIMESTAMP+30 and propagationFlag set to false!
```

```
3. Using the third session (client_2), check out “1.txt” and
store it as “1_copy2.txt”.
```

```
s2dr:client_2> check-out 1.txt 1_copy2.txt
Document 1.txt was saved to /s2dr/workspaces/client_2/1_copy2.txt
```

```
4. After the 30-second delegation made by the owner expires,
using the third session (client_2), check out “1.txt” and
store it as “1_copy3.txt”.
```

```
s2dr:client_2> check-out 1.txt 1_copy2.txt
You don't have checking-out permission.
```

5. After the 60-second delegation made by the second client expires, using the third session (client_2), check out “1.txt” and store it as “1_copy4.txt”.

```
s2dr:client_2> check-out 1.txt 1_copy4.txt
You don't have checking-out permission.
```

---

6. Terminate all sessions.

```
s2dr:client_0> terminate-session
s2dr:client_1> terminate-session
s2dr:client_2> terminate-session
```