“use strict”;

Let expect = require(‘expect.js’);

Const { Keychain } = require(‘../password-manager’);

Function expectReject(promise) {

Return promise.then(

() => Promise.reject(new Error(‘Expected failure’)),

() => {}

);

}

Describe(‘Password manager’, function() {

This.timeout(5000);

Let password = “password123!”;

Let kvs = {

“service1”: “value1”,

“service2”: “value2”,

“service3”: “value3”

};

Describe(‘functionality’, function() {

It(‘inits without an error’, async function() {

Await Keychain.init(password);

});

It(‘can set and retrieve a password’, async function() {

Let keychain = await Keychain.init(password);

Let url = ‘www.stanford.edu’;

Let pw = ‘sunetpassword’;

Await keychain.set(url, pw);

Expect(await keychain.get(url)).to.equal(pw);

});

It(‘can set and retrieve multiple passwords’, async function() {

Let keychain = await Keychain.init(password);

For (let k in kvs) {

Await keychain.set(k, kvs[k]);

}

For (let k in kvs) {

Expect(await keychain.get(k)).to.equal(kvs[k]);

}

});

It(‘returns null for non-existent passwords’, async function() {

Let keychain = await Keychain.init(password);

For (let k in kvs) {

Await keychain.set(k, kvs[k]);

}

Expect(await keychain.get(‘www.stanford.edu’)).to.be(null);

});

It(‘can remove a password’, async function() {

Let keychain = await Keychain.init(password);

For (let k in kvs) {

Await keychain.set(k, kvs[k]);

}

Expect(await keychain.remove(‘service1’)).to.be(true);

Expect(await keychain.get(‘service1’)).to.be(null);

});

It(‘returns false if there is no password for the domain being removed’, async function() {

Let keychain = await Keychain.init(password);

For (let k in kvs) {

Await keychain.set(k, kvs[k]);

}

Expect(await keychain.remove(‘www.stanford.edu’)).to.be(false);

});

It(‘can dump and restore the database’, async function() {

Let keychain = await Keychain.init(password);

For (let k in kvs) {

Await keychain.set(k, kvs[k]);

}

Let data = await keychain.dump();

Let contents = data[0];

Let checksum = data[1];

Let newKeychain = await Keychain.load(password, contents, checksum);

// Verify that restored data matches original

For (let k in kvs) {

Expect(await newKeychain.get(k)).to.equal(kvs[k]);

}

});

It(‘fails to restore the database if checksum is wrong’, async function() {

Let keychain = await Keychain.init(password);

For (let k in kvs) {

Await keychain.set(k, kvs[k]);

}

Let data = await keychain.dump();

Let contents = data[0];

Let fakeChecksum = ‘3GB6WSm+j+jl8pm4Vo9b9CkO2tZJzChu34VeitrwxXM=’;

Await expectReject(Keychain.load(password, contents, fakeChecksum));

});

It(‘returns false if trying to load with an incorrect password’, async function() {

Let keychain = await Keychain.init(password);

For (let k in kvs) {

Await keychain.set(k, kvs[k]);

}

Let data = await keychain.dump();

Let contents = data[0];

Let checksum = data[1];

Try {

Let result = await Keychain.load(“fakepassword”, contents, checksum);

// If Keychain.load doesn’t throw an error, check the result

Expect(result).to.be(false);

} catch (error) {

// If Keychain.load throws an error, ensure it’s the expected error

Expect(error).to.be.a(Error); // Corrected usage for expect.js

}

});

}); // Close the ‘functionality’ describe block

Describe(‘security’, function() {

// Test to ensure domain names and passwords are not stored in clear text

It(“doesn’t store domain names and passwords in the clear”, async function() {

Let keychain = await Keychain.init(password);

Let url = ‘www.stanford.edu’;

Let pw = ‘sunetpassword’;

Await keychain.set(url, pw);

Let data = await keychain.dump();

Let contents = data[0];

// Ensure contents do not contain sensitive information

Expect(contents).not.to.contain(password);

Expect(contents).not.to.contain(url);

Expect(contents).not.to.contain(pw);

});

// This test won’t be graded directly – it just exists to make sure your

// dump includes a kvs object with all your urls and passwords, because

// we will be using that in other tests.

It(‘includes a kvs object in the serialized dump’, async function() {

Let keychain = await Keychain.init(password);

For (let i = 0; i < 10; i++) {

Await keychain.set(String(i), String(i));

}

Let data = await keychain.dump();

Let contents = data[0];

Let contentsObj = JSON.parse(contents);

Expect(contentsObj).to.have.key(‘kvs’);

Expect(contentsObj.kvs).to.be.an(‘object’);

Expect(Object.getOwnPropertyNames(contentsObj.kvs)).to.have.length(10);

});

});

});