Hands on project report

Security mechanism exploration with an implementation of two-factor authentication

Kalsang Sherpa & Ha Tran

CPSC 385 Computer Security

Professor: Ewa Syta

Abstract

Nowadays, computer security has become more important than ever. Businesses, whether large or small, all want to protect their resources and data against increasingly sophisticated and well-planned hacking methods. Companies that have to deal with users registrations and sign-in all have the same security concern: users passwords. When users choose passwords that are easy to guess or unintentionally disclose them to other people, the passwords become a security weakness that can be exploited to create damage to assets. From users' point of view, we also want to protect our information in a secured but convenient way so that logging into our accounts does not require too much effort. Directly involved in this process day by day when logging into different accounts, we, for this project, want to explore different security mechanisms with a focus on two-factor authentication. We provide an extra layer of protection to verify users identity by requiring them to enter a one-time pad sent to their phones each time they log in in addition to entering their passwords as usual. We explore some librarie to implement hashing methods suitable for the scope of the project, experiment with different APIs to send text messages to users phones, and store the hashed passwords in the database. Thus, in addition to exploring with different mechanisms for security, we build a program that authenticates users by 2 factors: knowledge-based authentication by passwords and possession-based authentication by cell phones.

Acknowledgments

Thank you to Professor Syta for giving us the opportunity to work on a hands on project to be more skilled in using security mechanisms.

Table of Contents

Chapter 1: **Introduction**

Nowadays, using passwords as the only way to check users identity has become a weakness of security for many systems and websites for a number of reasons. First of all, many people today still use extremely predictable passwords. According to the website Gizmodo, the most popular passwords of 2017 are 123456, password, 12345678, qwerty, and 12345. These types of passwords make it incredibly easy for hackers to hack into these accounts and steal information. Some people even make the terrible mistake of using only one username and password for all different kinds of accounts. Even if this password is not an easy-to-guess one, if the hackers find out the passwords, all types of accounts from mails, social networks to banks will be compromised. Some users create long and complicated passwords to make it more difficult for hackers to hack them, but then they write the passwords down a piece of paper so that they will not forget them. If somebody else sees this piece of paper and copy them down, their difficult passwords are no longer safe. Therefore, using only passwords to prove users identity can be risky in a number of circumstances.

In security, authentication is the process of determining whether or not someone has the right to access a system or an account. There are three types of factors for authentication of users identities. The first type is knowledge-based authentication method, which requires users to remember a piece of information such as passwords, pin codes, or signatures to gain access to their accounts. This piece of information is supposed to be confidential to everybody but the owner of the account. Passwords are the most common type of authentication so far because they are the first choice of authentication. They are also free and convenient. The second authentication factor is possession-based, which is usually associated with a hardware device such as cell phones and tokens that only the right owner of the account should have access to. If cell phones are used as the second factor, then usually a one-time pad will be sent to the phone number of the user and he/she will enter this pad code to verify the second part of signing in. Token is a small device that can be carried around conveniently that has a unique PIN number on it, which may changes through time. Users simply put in this PIN numbers to log in. The last factor is biometric-based authentication. In order to gain access to the account, the owners must prove their identity through their own biological features such as their faces, fingerprints and so on.

One way to address the vulnerability of using passwords as the only authentication (one-factor authentication) is to use two-factor authentication to add an extra layer of security over users accounts. Two-factor authentication (2FA), accordingly, is the method that requires users to present two different factors for authentication. For example, if the first factor of authentication is passwords, which are knowledge-based, then the second factor of authentication has to be possession-based or biometric-based. Two-factor authentication is different from two-step authentication, which is the method where users use the same type of authentication to validate themselves. For instance, users first use a password they remember to complete the first step, then use the secret code sent to their email to complete the second step of verification. Emails and passwords are both knowledge-based verifications because emails are not associated to a single device but they can be opened in any device. Two-factor authentication is stronger and much more secure than the traditional one-factor authentication that only requires users passwords.

Many websites and systems nowadays use two-factor authentication to identify their users. Google uses passwords as the first factor and a PIN code sent to mobile phones as the second factor authentication. BesToken uses two-factor authentication through passwords and a smart card chip integrated USB token. The token is able to both generate and store users information such as passwords and keys [5]. Similarly, SecurID from RSA uses a token for authentication through an RSA Authentication Manager server [6]. Each token is used to generate a pseudo random number.

For the scope of this project, we use knowledge-based authentication - passwords - for the first factor, and possession-based authentication - sending a one-time pad to a cell phone - for the second factor of the method. This unique one-time pad is generated randomly and usable for a short, limited amount of time. Mobile phones are common devices that almost everyone carries with themselves at all times so we will use mobile phones instead of cards or tokens as the second factor authentication for our project. If the users do not enter the pad within the allowable amount of time, a new pad is sent to their phone numbers if they request another one. SMS authentication is the most realistic and achievable approach for the scope of the project. Using mobile phones is also cost-effective because users do not have to spend money and effort in buying, exchanging, and maintaining cards or tokens. A drawback of using mobile phones as second-factor authentication is that NIST has advised against using them because nowadays, it is possible and in some countries, very easy to duplicate sim cards. Out main goals were to learn and use one of the API among Twilio Copilot. In conclusion, this project explores different security mechanisms with a clear implementation of the two-factor authentication using cell phones as the second factor.

Chapter 2: **Background Research**

2.1 One time pad

The one-time pad algorithm is among the simplest cryptography and is considered by some to be unbreakable. It is an exclusive OR between the message (to be encrypted) and the pad (a random key - sequence of bits). As for the principles of one time pad first, the key must be random , second, parts of the key that have already been used to do encryption must not be available for other encryption. The key (Pad) must be a sequence of random bits as long as the message to be encrypted. The sender exclusive-OR's the message with the pad and sends the result through a communication channel. The one time requirement that makes it unbreakable and difficult at the same time is that after use, the sender must get rid of part of the pad, and not use it again. At the other end of the communication, the receiver must have an identical copy of the pad. The receiver decrypts the cipher text to obtain the original message by doing an exclusive-OR of the incoming cipher with its copy of the pad. The receiver should also destroy the pad after use.

2.2 Hashing password

To proceed with two factor authentication we need a way to store the user credentials in the database for future comparisons but storing passwords on the server side for authentication is a difficult task. Hashing makes password storage secure. To store a password it is secure to transform it into data that cannot be converted back to the original password. This process is called hashing [12].

- The user creates an account.
- Their password is hashed and stored in the database.
- When the user attempts to login, the hash of the password they entered is checked against the hash of their real password (retrieved from the database).
- If the hashes match, the user is granted access. If not, the user is told they entered invalid login credentials.

In cryptography, a hash function is a mathematical algorithm that maps data of any size to a bit string of a fixed size. We can refer to the function input as message or simply as input. The fixed-size string function output is known as the hash or the message digest. It is practical and easy to compute the hash, but impossible to re-generate the original input if only the hash value is known. Common hashing algorithms include Message Digest (MDx) algorithms, such as MD5, and Secure Hash Algorithms (SHA), such as SHA-1 and the SHA-2 family that includes the widely used SHA-256 algorithm.

2.2.1 To store a password

- Generate a long random salt using a CSPRNG.
- Ass the salt to the password and hash it with a standard password hashing function like bcrypt, scrypt, or PBKDF2.
- Save both the salt and the hash in the user's database record.

2.2.2 To validate a password

- Retrieve the user's salt and hash from the database.
- Add the salt to the given password and hash it using the same hash function.
- Compare the hash of the given password with the hash from the database. If they match, the password is correct. Otherwise, the password is incorrect.

Chapter 3: **Design and Implementation**

3.1 System design overview

Class	Instance variables	Methods
User	Username Password Phone number	Constructor Get variables Set variables
UserRepository	Hashmap of users	Hash function Create, delete, update, retrieve user
PhoneVerification	Hashmap of users	Send one-time pad to phones
Login	Hashmap of users	Validate password Validate one-time pad
Generator	Length Uppercase Lowercase Special character	Create one-time pad

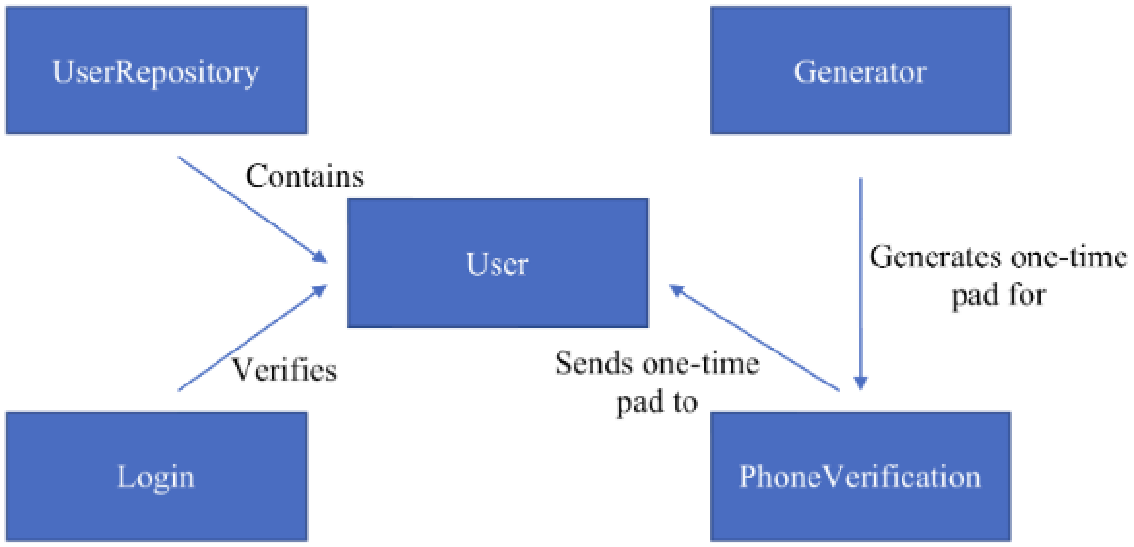


Figure 3.1: Showing connection between classes.

In this project, we use Java for the programming language, NetBeans for the development environment, Firebase for the data storage online, JavaFX for GUI. We will learn Bouncy Castle library for cryptography. We will experiment with three different APIs to send SMS to users' phones: Twilio Copilot, Telegram, and Bandwidth. Last but not least, we will explore different hashing methods for passwords.

Chapter 4: **Java Classes**

4.0.1 User

This class implements serialization. Serialization is a mechanism of converting the state of an object into a byte stream. It contains get and set methods with three instance variables.

- name
- password
- phone number

This class is called in *UserRepository* class.

4.0.2 UserRepository

AddUser method returns a boolean by passing a parameter from the User class gets a new user. It invokes method generateID() which and serID(). This class similarly contains methods to modify, delete and retrieve user.

4.0.3 Hash

Creates a hashmap and calls java classes *FileConverter.java* and *BasicFileConverter.java*. It reads the user information/list from the data base. A String password is passed to method *hash()* which returns a password hashed as a string. The library MessageDigest is used in this method where method getInstance is invoked and parameter "SHA-512" is passed to digest the message using SHA-512.

4.0.4 Generator

Generates a random string salt to ensure no two accounts have the same salt.

4.0.5 PhoneVerification

Twilio API is used to send SMS to mobile phones with verification code.

4.0.6 BasicFileConverter

Creating a hashmap of users and pad to write store it in database.

4.0.7 FileConverter

Interface for *BasicFileConverter.java*.

4.0.8 LogIn

Class containing hashmap of users to verify username and password

4.0.9 Tester Classes

- UserTester.java
- UserRepositoryTester.java
- GeneratorTester.java
- PhoneVerificationTester.java
- LogInTester.java

Chapter 5: **Libraries used**

5.0.1 java.io.*

The Basic Input Output From the user to java application

5.0.2 java.util.*

Contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes (a string tokenizer, a random-number generator, and a bit array) [13].

5.0.3 java.security.MessageDigest

The Java Cryptographic services include signature, message digest, cipher, mac and key stores. The MessageDigest class supports message digest algorithms MD2, MD5,SHA-1, SHA-256,SHA-384 and SHA-512. SHA-256 is a 256-bit hash function to provide 128 bits of security against collision attacks. SHA-512 is a 512 bit hash function to provide 256 bits of security. A 384-bit hash is obtained by truncating the SHA-512 output [15]. This MessageDigest class provides applications the functionality of a message digest algorithm, such as SHA-1 or SHA-256. Message digests are secure one-way hash functions that take arbitrary-sized data and output a fixed-length hash value. [14].

5.0.4 java.security.NoSuchAlgorithmException

NoSuchAlgorithmException extends GeneralSecurityException This exception is thrown when a particular cryptographic algorithm is requested but is not available in the environment.

5.0.5 java.security.SecureRandom

This class provides a strong random number generator. Additionally, SecureRandom must produce non-deterministic output. Therefore any seed material passed to a SecureRandom object must be unpredictable, and all SecureRandom output sequences must be cryptographically strong.Callers may also invoke the generateSeed method to generate a given number of seed bytes .Method nextbytes() generates a user-specified number of random bytes.

5.0.6 java.io.Serializable

Serializability of a class is enabled by the class implementing the *java.io.Serializable* interface. Implementing classes with serialzed interface makes the class serialized. The serialization interface has no methods or fields and serves only to identify the semantics of being serializable.

5.0.7 java.io.BufferedReader

This librarly allows a text to be read from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines. Without buffering, each invocation of read() or readLine() could cause bytes to be read from the file, converted into characters, and then returned, which can be very inefficient.

5.0.8 java.io.FileReader

FileReader is meant for reading streams of characters. For reading streams of raw bytes, consider using a FileInputStream. The constructors of this class assume that the default character encoding and the default byte-buffer size are appropriate. To specify these values yourself, construct an InputStreamReader on a FileInputStream.

5.0.9 java.io.IOException

Signals that an I/O exception of some sort has occurred.

5.0.10 java.util.logging.Level

Specifies the severity level of an event.

5.0.11 java.util.logging.Logger

This library logs messages for an application by using a logger object. Logger names can be arbitrary strings, but they should normally be based on the package name or class name of the logged component, such as java.net or javax.swing. In addition it is possible to create "anonymous" Loggers that are not stored in the Logger namespace.

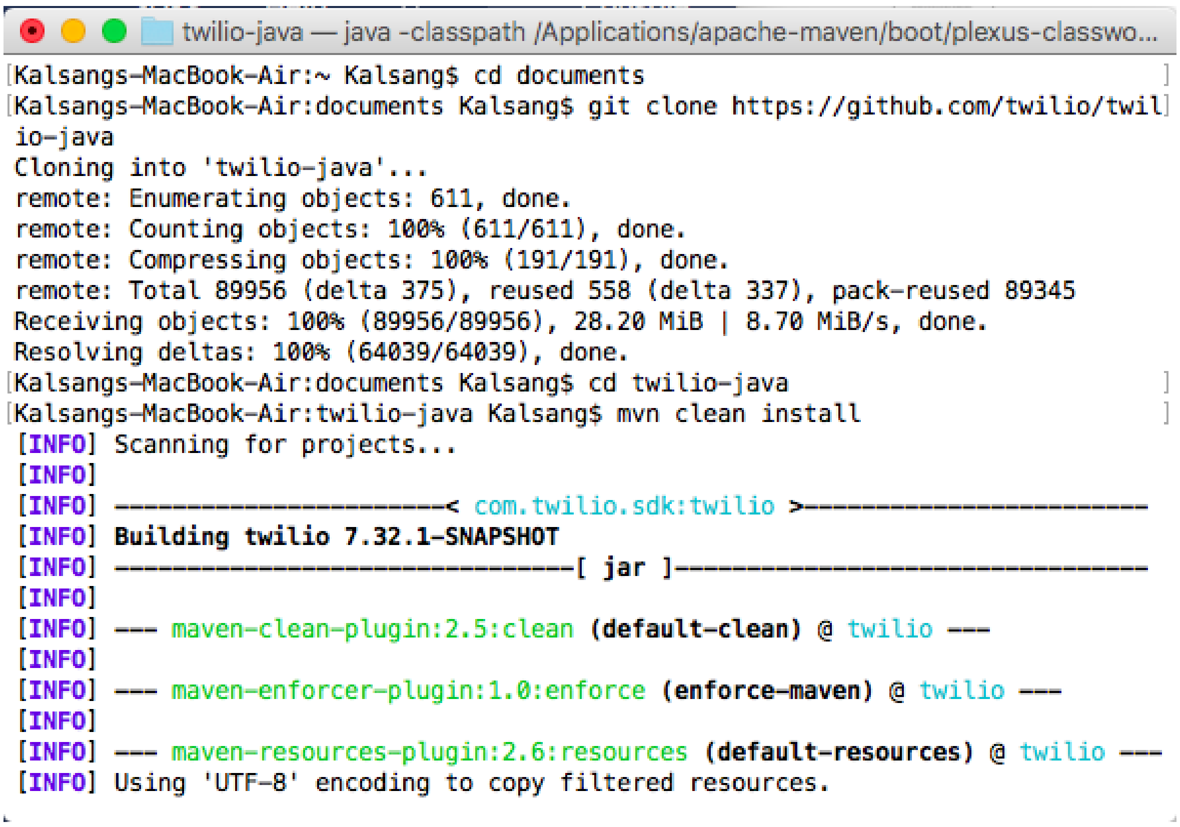
Chapter 6: **Tools used**

6.1 Twilio

Twilio is a developer platform for communications, providing APIs to add capabilities like voice, video, and messaging to their applications. It is a cloud based service that enables powerful communication between mobile devices, applications, services, and systems throughout the business in order to bridge the gap between conventional communication. This enables businesses to provide the right communications experience for their customers. Behind Twilio APIs is a Super Network, a software layer that connects and optimizes communications networks around the world.

6.1.1 Using Twilio API to send SMS

The Twilio Java SDK helps us interact with the Twilio API from the Java application. The recommended method for installing the SDK is with a build automation tool, like Maven or Gradle. Here we instal Maven for this project.



```
twilio-java — java -classpath /Applications/apache-maven/boot/plexus-classwo...
Kalsangs-MacBook-Air:~ Kalsang$ cd documents
Kalsangs-MacBook-Air:documents Kalsang$ git clone https://github.com/twilio/twil
io-java
Cloning into 'twilio-java'...
remote: Enumerating objects: 611, done.
remote: Counting objects: 100% (611/611), done.
remote: Compressing objects: 100% (191/191), done.
remote: Total 89956 (delta 375), reused 558 (delta 337), pack-reused 89345
Receiving objects: 100% (89956/89956), 28.20 MiB | 8.70 MiB/s, done.
Resolving deltas: 100% (64039/64039), done.
Kalsangs-MacBook-Air:documents Kalsang$ cd twilio-java
Kalsangs-MacBook-Air:twilio-java Kalsang$ mvn clean install
[INFO] Scanning for projects...
[INFO]
[INFO] -----------------------< com.twilio.sdk:twilio >------------------------
[INFO] Building twilio 7.32.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ twilio ---
[INFO]
[INFO] --- maven-enforcer-plugin:1.0:enforce (enforce-maven) @ twilio ---
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ twilio ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
```

Figure 6.1: Commands to install Twilio API

6.1.2 Twilio libraries

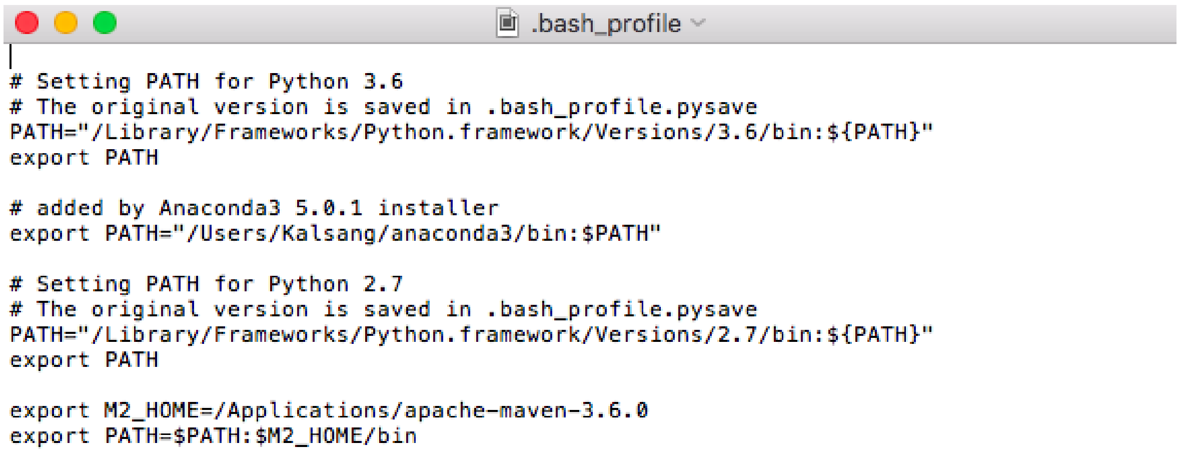
- import com.twilio.Twilio;
- import com.twilio.rest.api.v2010.account.Message;
- import com.twilio.type.PhoneNumber;

6.2 Apache Maven Project

It was important to install maven to use Twilio. Maven dynamically downloads Java libraries and Maven plug-ins from one or more repositories and stores them in a local cache. Mavens primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period of time. In order to attain this goal there are several areas of concern that Maven attempts to deal with:

Making the build process easy Providing a uniform build system Providing quality project information Providing guidelines for best practices development Allowing transparent migration to new features

6.2.1 Installing Apache Maven Project

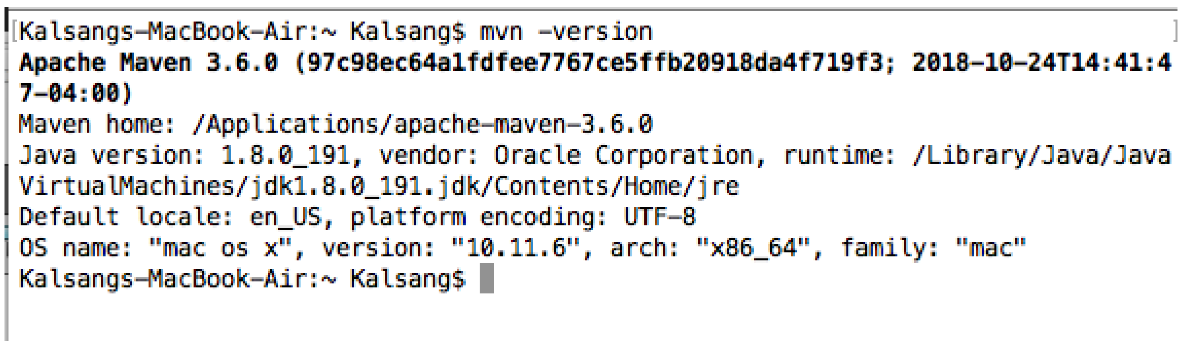


```
# Setting PATH for Python 3.6
# The original version is saved in .bash_profile.pysave
PATH="/Library/Frameworks/Python.framework/Versions/3.6/bin:${PATH}"
export PATH

# added by Anaconda3 5.0.1 installer
export PATH="/Users/Kalsang/anaconda3/bin:$PATH"

# Setting PATH for Python 2.7
# The original version is saved in .bash_profile.pysave
PATH="/Library/Frameworks/Python.framework/Versions/2.7/bin:${PATH}"
export PATH

export M2_HOME=/Applications/apache-maven-3.6.0
export PATH=$PATH:$M2_HOME/bin
```

Figure 6.2: With the command line **$ open -e bash.profile** we add the export path to directpry containing maven file

6.3 NetBeans

NetBeans is an open-source integrated development environment (IDE) for developing with Java, PHP, C++, and other programming languages. It also referred to as a platform of modular components used for developing Java desktop applications. [16]



```
Kalsangs-MacBook-Air:~ Kalsang$ mvn -version
Apache Maven 3.6.0 (97c98ec64a1fdfee7767ce5ffb20918da4f719f3; 2018-10-24T14:41:4
7-04:00)
Maven home: /Applications/apache-maven-3.6.0
Java version: 1.8.0_191, vendor: Oracle Corporation, runtime: /Library/Java/Java
VirtualMachines/jdk1.8.0_191.jdk/Contents/Home/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "mac os x", version: "10.11.6", arch: "x86_64", family: "mac"
Kalsangs-MacBook-Air:~ Kalsang$
```

Figure 6.3: Command line **$ mvn** -**version** output

Chapter 7: **Result**

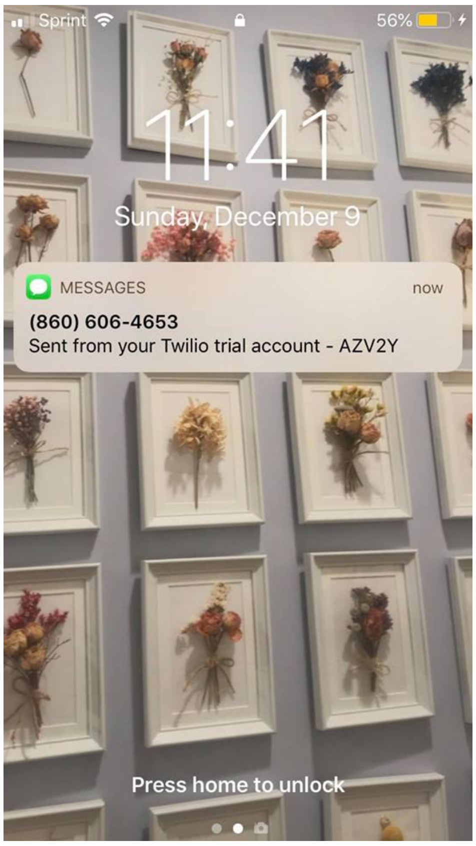
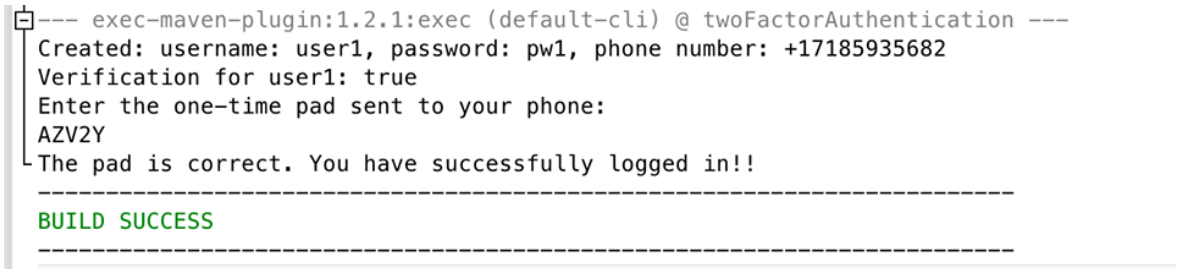

Sprint 56%
11:41
Sunday, December 9
MESSAGES now
(860) 606-4653
Sent from your Twilio trial account - AZV2Y
Press home to unlock

Figure 7.1: SMS with verification code



```
--- exec-maven-plugin:1.2.1:exec (default-cli) @ twoFactorAuthentication ---
Created: username: user1, password: pw1, phone number: +17185935682
Verification for user1: true
Enter the one-time pad sent to your phone:
AZV2Y
The pad is correct. You have successfully logged in!!
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

Figure 7.2: Verification code input

Bibliography

[1] Schneier, B. (2005). *Two-factor authentication*. Association for Computing Machinery. Communications of the ACM, 48(4), 136.

[2] Eldefrawy, M. H., Khan, M. K., K., Elkamchouchi, H. (2011, May 18). *Mobile one-time passwords: Two-factor authentication using mobile phones.*

[3] Akram S, Misbahuddin M, Varaprasad G. `A Usable and Secure Two-Factor Authentication Scheme.1` Information Security Journal: A Global Perspective. 2012;21(4):169-182. doi:10.1080/19393555.2011.629340

[4] Zhang, J., Tan, X., Wang, X., Yan, A., Qin, Z. (2018). *T2FA: Transparent Two-Factor Authentication.* . Access, IEEE, 6, 32677-32686.

[5] Aloul, F., Zahidi, S., El-Hall, W. (2009, June 05). *Two factor authentication using mobile phones.*. Access, IEEE, DOI: 10.1109/AICCSA.2009.5069395.

[6] *Two-Factor Authentication Overview.*
`http://help.sonicwall.com/help/sw/eng/8112/8/0/0/content/Chapter2_Overview.03.23.html`

[7] Stamp, M. (2011). *Information security principles and practices,2nd Edition*

[8] Buke, R. (2018, September 10). *Top 10 Voice SMS APIs to Send Calls Text Messages — RapidAPI.*
`https://blog.rapidapi.com/sms-apis-send-texts/`

[9] *Sending and receiving SMS From Java using the Ozeki Java SMS SDK.*
`http://www.ozekisms.com/index.php?owpn=584.`

[10] Amin, A., Haq, I. U., Nazir, M. (2017, July). *Two factor Authentication.*International Journal of Computer Science and Mobile Computing, 6(7), ijcsmc, 5-8.

[11] Chen, A. Q., Goh, W. (2015, June). *Two Factor Authentication Made Easy.*

[12] Auth0 (2018, April 25). *Hashing Passwords: One-Way Road to Security*
`https://auth0.com/blog/hashing-passwords-one-way-road-to-security/`

[13] Quora *Article TitleWhy, in Java, is there the line "import Java.util.*"?*
`https://www.quora.com/Why-in-Java-is-there-the-line-import-Java-util-*`

[14] (2018, October)
`https://docs.oracle.com/javase/7/docs/api/java/security/MessageDigest.html`

[15] (2015, February) *java.security.MessageDigest Example ()*
`https://examples.javacodegeeks.com/core-java/security/java-security-messagedigest-example/`

[16] *What Is NetBeans? - Definition from Techopedia.*
`www.techopedia.com/definition/24735/netbeans.`