- Checker Framework:
- Main Site
- Manual
- Discussion List
- Issue Tracker
- Source Code
- Tutorial

Checker Framework Tutorial

Previous Finding a Security Error, Download Example Sources

Writing an Encryption Checker *Optional*

This section of the tutorial is only for those who are interested in writing their own type-checkers. Please feel free to skip this section.

Although the Checker Framework ships with many checkers, you may wish to write your own checker because there are other run-time problems you wish to prevent. If you do not wish to write a new type-checker, feel free to skip this section of the tutorial.

Outline

1. An Encryption Checker

As an example, suppose that you wish to only allow encrypted information to be sent over the internet. To do so, you can write an Encryption Checker.

2. Write the type annotation definitions

For this example, the annotation definitions have already been written for you and appear in files `Encrypted.java`, `PossiblyUnencrypted.java`, and `PolyEncrypted.java`.

Compile the annotation definitions.

```
$ javacheck myqual/Encrypted.java myqual/PossiblyUnencrypted.java
myqual/PolyEncrypted.java
```

3. Run the Encryption Checker

The `@Encrypted` annotations have already been written in file EncryptionDemo.java. The default for

types without annotations is `@PossiblyUnencrypted`.

Invoke the compiler with the Subtyping Checker. Specify the @Encrypted, @PossiblyUnencrypted, and @PolyEncrypted annotations using the -Aquals command-line option, and add the Encrypted, PossiblyUnencrypted, and PolyEncrypted classfiles to the processor classpath:

```
$ javacheck -processor org.checkerframework.common.subtyping.SubtypingChecker -
Aquals=myqual.Encrypted,myqual.PossiblyUnencrypted,myqual.PolyEncrypted
encrypted/EncryptionDemo.java
encrypted/EncryptionDemo.java:21: error: [assignment.type.incompatible]
incompatible types in assignment.
        @Encrypted int encryptInt = (character + OFFSET) % Character.MAX_VALUE ;
                                                           ^
  found   : @PossiblyUnencrypted int
  required: @Encrypted int
encrypted/EncryptionDemo.java:32: error: [argument.type.incompatible] incompatible
types in argument.
        sendOverInternet(password);
                         ^
  found   : @PossiblyUnencrypted String
  required: @Encrypted String
2 errors
```

4. Suppress the First Error

The first error needs to be suppressed, because the string on the left is considered "encrypted" in this encryption scheme. All `@SuppressWarnings` should have a comment explaining why suppressing the warning is the correct action. See the correction below.


```
@SuppressWarnings("encrypted")  // this is the encryption algorithm
private @Encrypted char encryptCharacter(char character) {
```


5. Re-run the Encryption Checker

You will see the following error:


```
$ javacheck -processor org.checkerframework.common.subtyping.SubtypingChecker -
Aquals=myqual.Encrypted,myqual.PossiblyUnencrypted,myqual.PolyEncrypted
encrypted/EncryptionDemo.java
encrypted/EncryptionDemo.java:34: error: [argument.type.incompatible] incompatible
types in argument.
        sendOverInternet(password);
                         ^
  found   : @PossiblyUnencrypted String
  required: @Encrypted String
1 error
```

This is a real error, because the programmer is trying to send a password over the Internet without encrypting it first.

6. Correct the Second Error

The password should be encrypted before it is sent over the Internet. The correction is below.

```
void sendPassword() {
    String password = getUserPassword();
    sendOverInternet(encrypt(password));
}
```

7. Re-run the Encryption Checker

```
$ javacheck -processor org.checkerframework.common.subtyping.SubtypingChecker -
Aquals=myqual.Encrypted,myqual.PossiblyUnencrypted,myqual.PolyEncrypted
```

`encrypted/EncryptionDemo.java`

There should be no errors.

8. Learn more about writing your own checker

For further explanations, see the Checker Framework manual, chapter How to create a new checker.

The end of the Checker Framework Tutorial Return to the main page of the Tutorial.