Java: Secure Coding Practices

Brandon Thompson

October 29, 2019

1 Ensure that sensitive data is kept secure

```
openFileOutput(”someFile”, MODE_WORLD_READABLE);
```
Problem: creates a file that is world readable. Any application could read the file and access any data.
```
openFileOutput(”someFile”, MODE_PROVATE);
```

2 Detect and Handle File Related Errors

```
File file = new File(”ABCfile”);
file.delete()
```
Problem: delete a specified file but gives no indication of it ssuccess. `File.delete()` throws a `SecurityException`

```
File file = new File
```

3 Generate Strong Random Numbers

```
Random number = new Random();
//...
for (int i = 0; i < 20; i++) {
int n = number.nextInt(21);
System.out.println(n);
```
Produce an indentical sequence

4 Dont use the Object.equals() method to compare arrays

```
int[] arr1 = new int[20];
int[] arr2 = new int[20];
System.out.println(arr.equals(arr2));
```

```
int[] arr1
```
Problem: Object.equals() compares only

5 Validate Method Arguments

```
private Object myState = null;

void setState(Object state) {
myState = state;
}
```

6 Ensure that division and remainder operations do no result in divide by zero errors.

```
long num1, num2, result;
result = num1 / num2;
result = num1 % num2;
```

Problem: can result in a division-by-zero error.

```
long num1, num2, result
```

7 Do not attempt comparisons with NaN

```
double x = 0.0
```

8 Limit Accessibility of Fields

```
public class Widget {
public int total;
...
public void add() {...}
```