Quo pertentas, OSS?

How Open Source can benefit from well-crafted Tests

Björn Kimminich
Web: http://kimminich.de
Twitter: @bkimminich
Let's start with some code…

```java
public class Server {

    private String memory;

    public Server(String username, String password) {
        this.memory = this.hashCode() + ";user=" + username + ";password=" + password + ";" + this.getClass().getName();
    }

    public String heartbeat(String payload, int length) {
        memory = "pl=" + payload + memory;
        return memory.substring(3, 3+length);
    }
}
```
...and a corresponding unit test!

```java
@Test
public void testHeartbeat() throws Exception {
    Server server = new Server("bjoern.kimminich", "s3cret123");
    String payload = "1100101";
    String reply = server.heartbeat(payload, payload.length());
    assertEquals(payload, reply);
}
```
It passes with flying colors...
... and achieves 100% code coverage!
Nothing could possibly go wrong!
How about adding another test?

```java
@Test
public void testHeartbleed() throws Exception {
    Server server = new Server("bjoern.kimminich", "s3cret123");
    String payload = "1";
    String reply = server.heartbeat(payload, 64);
    assertEquals(payload, reply);
}
```
Oops!
Finding Bugs in Open Source Software
Code Reviews

Pair Programming
Infeasible with remote development
Occasionally during Hackathons

Peer Review
Developers review each other
Hard to organize properly

Committer Review
Not everyone has commit rights
Senior developers review contributions before merge into master

How to Make a Good Code Review

Rule 1: Try to find at least something positive.
## Static Code Analysis

### Popular Open Source Tools

<table>
<thead>
<tr>
<th>FindBugs</th>
<th>CheckStyle</th>
<th>PMD</th>
<th>Sonar</th>
</tr>
</thead>
</table>

Find code smells and potential programming errors…

…but miss a lot as well…

…or produce false positives

Some commercial Tools might be more powerful…

…but are typically not affordable for OSS projects

Cartoon: [Geek & Poke]( 書籍 www.geekandpoke.com)
Testing

**Test Types**

- Unit Tests
- Integration Tests
- GUI Tests
- Manual Tests
- Load/Stress Tests
- Penetration Tests

*PROJECT MANAGEMENT MADE EASY*

*Cartoon: Geek & Poke*
Best vs. Bad Practices for Testing
Test Pyramid

- Manual Tests
- GUI Tests
- Integration Tests
- Unit Tests

Source: WatirMelon
Test Ice-Cream Cone

- Unit Tests
- Integration Tests
- GUI Tests
- Manual Tests

Source: WatirMelon
Happy Path Testing
Testing Border & Exceptional Cases

```java
@Test(expected = IllegalArgumentException.class)
public void shouldThrowExceptionOnMissingBaseUrl() { URLResolver.resolveUrl(null, "notNull"); }

@Test(expected = IllegalArgumentException.class)
public void shouldThrowExceptionOnMissingRelativeUrl() { URLResolver.resolveUrl("notNull", null); }

@Test
public void shouldAppendRelativeUrlToBaseUrlHost() {
    assertThat(URLResolver.resolveUrl("http://www.abc.de", "/xy/z"), is("http://www.abc.de/xy/z"));
}

@Test
public void shouldInsertSlashBetweenBaseUrlAndRelativeUrlIfMissing() {
    assertThat(URLResolver.resolveUrl("http://www.abc.de", "xyz"), is("http://www.abc.de/xyz"));
}

@Test
public void shouldReplaceLastPartOfUrlPathFromBaseUrlWithRelativeUrl() {
    assertThat(URLResolver.resolveUrl("http://www.abc.de/x/x", "x/z"), is("http://www.abc.de/x/x/z"));
}
```
public void testAllMethods() throws Exception {
    // create a user to test Anonymous
    String accountName = ESAPI.randomizer().getRandomString(8, EncoderConstants.CHAR_ALPHANUMERIC);
    Authenticator instance = ESAPI.authenticator();
    String password = instance.generateStrongPassword();

    // Probably could skip the assignment here, but maybe someone had
    // future plans to use this. So will just suppress warning for now.
    //unused/
    User user = instance.createUser(accountName, password, password);

    // test the rest of the Anonymous user
    try {
        User.ANONYMOUS.addRole(null); } catch( RuntimeException e ) {}
    try {
        User.ANONYMOUS.addRoles(null); } catch( RuntimeException e ) {}
    try {
        User.ANONYMOUS.changePassword(null, null, null); } catch( RuntimeException e ) {}

    /* [...] */
    try {
        User.ANONYMOUS.getAccountName(); } catch( RuntimeException e ) {}
API Tests
Scenario Tests with BDD

```java
@Test
public void compressedResponseBodyShouldBeDeflatedIntoApiResponse() throws Exception {
    given(responseHeader.getHeader(Header.CONTENT_ENCODING)).willReturn(Header.GZIP);
    given(responseBody.getBytes()).willReturn(gzip(new byte[] {97, 98, 99}));

    ApiResponseSet response = ApiResponseConversionUtils.httpMessageToSet(0, message);

    assertThat(response.getValues(), hasEntry("responseBody", (Object)"abc"));
}

@Test
public void brokenCompressedResponseBodyShouldBeStoredAsStringRepresentationInApiResponse() {
    given(responseHeader.getHeader(Header.CONTENT_ENCODING)).willReturn(Header.GZIP);
    given(responseBody.getBytes()).willReturn(new byte[] {0,0,0});

    ApiResponseSet response = ApiResponseConversionUtils.httpMessageToSet(0, message);

    assertThat(response.getValues(), hasEntry("responseBody", (Object)responseBody.toString()));
}
```
Benefits of well-crafted Tests for OSS
Maintainability++

A suite of automated regression tests helps finding defects resulting from code changes.

New contributors do not have to fear touching old code…

…neither do long-time committers after a longer vacation!

Cartoon: Geek & Poke
External and Javadoc documentation tends to rot quickly and becomes obsolete or even misleading.

Tests that get outdated tend to break, so they have to be fixed resulting in updated documentation.

Well-written tests document the intended behavior of a class or component.

Even if the production code is hard to understand, a good test can help to fill this gap.
Specification++

Writing tests before the production code is even better than just documenting existing code.

Consequent TDD / BDD will let the Tests become the actual specification of the program's intended behavior.

Failing tests indicate that the specification is not met yet (or any more).

Cartoon: Geek & Poke
Contribution++

Well maintained, documented and tested projects are safer and more fun to contribute to.

Nobody likes working on an untested piece of unreadable code (especially in their free time).

Cartoon: Geek & Poke
Truck Factor++

How many project contributors could be fatally hit by a truck before the project perishes?

The lower the number, the more volatile the project as it relies on individual experts

The number can be increased by spreading knowledge and lowering entry barriers
Introducing Unit Tests to OWASP ZAP
# OWASP Zed Attack Attack Proxy (ZAP)

**Easy-to-use integrated penetration testing tool**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locates vulnerabilities in web applications</td>
<td>Helps building secure apps</td>
</tr>
</tbody>
</table>

**OWASP Flagship Project**

Programmed in Java with `javax.swing` UI
How to contribute to ZAP?

- Develop core features https://code.google.com/p/zaproxy/
- Develop addons https://code.google.com/p/zap-extensions/
- Help with translation https://crowdin.net/project/owasp-zap
- Promote ZAP https://code.google.com/p/zaproxy/wiki/ZapEvangelists
ZAP Truck Factor ≤2

<table>
<thead>
<tr>
<th>Name</th>
<th>Kudos</th>
<th>12 Month Commits</th>
<th>All Time Commits</th>
<th>5 Year Trend</th>
<th>Primary Language</th>
<th>First Commit</th>
<th>Last Commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simon Bennetts</td>
<td>5</td>
<td>693</td>
<td>1607</td>
<td></td>
<td>Java</td>
<td>over 3 years ago</td>
<td>6 days ago</td>
</tr>
<tr>
<td><a href="mailto:THCH...@gmail.com">THCH...@gmail.com</a></td>
<td>4</td>
<td>438</td>
<td>710</td>
<td></td>
<td>Java</td>
<td>over 2 years ago</td>
<td>6 days ago</td>
</tr>
<tr>
<td><a href="mailto:colm.p.o...@gmail.com">colm.p.o...@gmail.com</a></td>
<td>5</td>
<td>29</td>
<td>241</td>
<td></td>
<td>Java</td>
<td>almost 2 years ago</td>
<td>8 days ago</td>
</tr>
<tr>
<td>Cosmin Stefan Dobrin</td>
<td>8</td>
<td>127</td>
<td>186</td>
<td></td>
<td>Java</td>
<td>almost 2 years ago</td>
<td>23 days ago</td>
</tr>
<tr>
<td><a href="mailto:yha...@gmail.com">yha...@gmail.com</a></td>
<td>7</td>
<td>176</td>
<td>176</td>
<td></td>
<td>Java</td>
<td>18 months ago</td>
<td>26 days ago</td>
</tr>
<tr>
<td>Björn Kimmich</td>
<td>4</td>
<td>40</td>
<td>97</td>
<td></td>
<td>Java</td>
<td>over 1 year ago</td>
<td>30 days ago</td>
</tr>
<tr>
<td><a href="mailto:maw...@ymail.com">maw...@ymail.com</a></td>
<td>7</td>
<td>0</td>
<td>76</td>
<td></td>
<td>Java</td>
<td>about 3 years ago</td>
<td>over 1 year ago</td>
</tr>
<tr>
<td><a href="mailto:70pco...@gmail.com">70pco...@gmail.com</a></td>
<td>7</td>
<td>74</td>
<td>74</td>
<td></td>
<td>Java</td>
<td>4 months ago</td>
<td>8 days ago</td>
</tr>
<tr>
<td>desousa.vitor</td>
<td>7</td>
<td>0</td>
<td>67</td>
<td></td>
<td>Java</td>
<td>about 2 years ago</td>
<td>about 2 years ago</td>
</tr>
<tr>
<td><a href="mailto:leandro...@talsoft.co">leandro...@talsoft.co</a></td>
<td>7</td>
<td>0</td>
<td>59</td>
<td></td>
<td>Java</td>
<td>over 1 year ago</td>
<td>over 1 year ago</td>
</tr>
<tr>
<td>a.c.neumann</td>
<td>7</td>
<td>0</td>
<td>57</td>
<td></td>
<td>Java</td>
<td>over 3 years ago</td>
<td>about 1 year ago</td>
</tr>
</tbody>
</table>

Source: Ohloh
Starting from zero Unit Tests

No Unit Tests

Some JUnit-based Integration tests
Separate Test Project

**zaproxy**
OWASP ZAP: An easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

**zap-extensions**
OWASP ZAP Extensions

**zaproxy-test**
Test code for the OWASP Zed Attack Proxy
package ch.cmac.extension.util;

public class EncodingUnitTest {
    // Test
    public void shouldConvertBytesToCorrectBase64String() {
        assertEquals(Encoding.base64Encode("Hello World"), "aHR0cDovL3N0YW5jZS5icmI=");
    }
    // Test
    public void shouldConvertBase64StringToCorrectBytes() {
        assertEquals(Encoding.base64Decode("aHR0cDovL3N0YW5jZS5icmI="), "Hello World");
    }
    // Test
    public void shouldConvertStringToCorrectMD5Hash() {...}
    // Test
    public void shouldConvertStringToCorrectSHA1Hash() {...}
    // Test
    public void shouldConvertStringToCorrectBlowfish() {...}
Adding some more Show Cases
Separation into Test Suites
Providing Test Guidelines

Types of Tests

Test Suites

Test Libraries

Naming Conventions

Behavior Driven Development

Code Quality

Code Coverage
Pull vs. Push

Photos: One Man Think Tank
Measure Code Coverage

<table>
<thead>
<tr>
<th>Element</th>
<th>Class, %</th>
<th>Method, %</th>
<th>Line, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.parroproxy.paros</td>
<td>100% (1/1)</td>
<td>10% (3/30)</td>
<td>8% (30/394)</td>
</tr>
<tr>
<td>org.parroproxy.paros.model</td>
<td>100% (1/1)</td>
<td>36% (7/19)</td>
<td>37% (36/95)</td>
</tr>
<tr>
<td>org.parroproxy.paros.spider</td>
<td>100% (1/1)</td>
<td>81% (9/11)</td>
<td>84% (11/13)</td>
</tr>
<tr>
<td>org.parroproxy.paros.watch</td>
<td>100% (1/1)</td>
<td>53% (8/15)</td>
<td>65% (15/23)</td>
</tr>
<tr>
<td>org.parroproxy.paros.users</td>
<td>100% (1/1)</td>
<td>57% (13/20)</td>
<td>69% (12/17)</td>
</tr>
<tr>
<td>ch.cscn.extension.util</td>
<td>100% (1/1)</td>
<td>92% (12/13)</td>
<td>80% (12/15)</td>
</tr>
<tr>
<td>org.parroproxy.paros.class</td>
<td>100% (1/1)</td>
<td>75% (3/4)</td>
<td>85% (17/20)</td>
</tr>
<tr>
<td>org.parroproxy.paros.network</td>
<td>100% (1/1)</td>
<td>54% (6/11)</td>
<td>41% (1/7)</td>
</tr>
<tr>
<td>org.parroproxy.paros.model</td>
<td>100% (1/1)</td>
<td>36% (1/8)</td>
<td>31% (1/3)</td>
</tr>
<tr>
<td>org.parroproxy.paros.spider</td>
<td>100% (1/1)</td>
<td>88% (24/27)</td>
<td>88% (27/30)</td>
</tr>
<tr>
<td>org.parroproxy.paros.authenticity</td>
<td>100% (1/1)</td>
<td>34% (16/47)</td>
<td>27% (50/184)</td>
</tr>
<tr>
<td>org.parroproxy.paros.control</td>
<td>100% (1/1)</td>
<td>75% (3/4)</td>
<td>85% (17/20)</td>
</tr>
<tr>
<td>org.parroproxy.paros.class</td>
<td>100% (1/1)</td>
<td>92% (12/13)</td>
<td>80% (12/15)</td>
</tr>
<tr>
<td>org.parroproxy.paros.network</td>
<td>100% (1/1)</td>
<td>75% (3/4)</td>
<td>85% (17/20)</td>
</tr>
<tr>
<td>ch.cscn.extension.util</td>
<td>100% (1/1)</td>
<td>92% (12/13)</td>
<td>80% (12/15)</td>
</tr>
<tr>
<td>org.parroproxy.paros.class</td>
<td>100% (1/1)</td>
<td>75% (3/4)</td>
<td>85% (17/20)</td>
</tr>
<tr>
<td>ch.cscn.extension.util</td>
<td>100% (1/1)</td>
<td>92% (12/13)</td>
<td>80% (12/15)</td>
</tr>
</tbody>
</table>
Move Tests close to Production Code

Unit Tests

Build Script

Integration Tests
Instant execution from IDE
Run all Tests during Continuous Build...

```xml
<target name="test" depends="compile">
  <!-- Run the JUnit tests -->
  <junit printsummary="yes" haltonerror="on">
    <classpath>
      <!-- [...] -->
    </classpath>
    <formatter type="plain"/>
    <formatter type="xml"/>
    <batchtest fork="yes" todir="results">
      <filesystem dir="${build}">
        <include name="**/*UnitTest.class"/>
        <exclude name="**/*Abstract*Test.class"/>
      </filesystem>
    </batchtest>
  </junit>
  <!-- [...] -->
</target>
```
...and let it fail when any tests fail!
Future: Adding a GUI Testing Framework

ZAP is very UI heavy which makes a lot of the code hard or impossible to unit test.

Right now there are no GUI Tests in place for ZAP.

Several free UI Testing Frameworks exist for Java Swing…

…unfortunately none is actively maintained any more.
Testing is a crucial part of Software Development

Good Tests are the better documentation

Tests can make a difference between a prospering and a dead-end OSS project
Thank you!

Björn Kimminich
Web: http://kimminich.de
Twitter: @bkimminich