Compile-time detection of machine image sniping

Martin Kellogg, University of Washington

Whose software is your cloud computer running?

Definition: a **machine image** is the set of software used to initialize a new cloud computer

Developers search repositories for machine images:

<table>
<thead>
<tr>
<th>Filters</th>
<th>Example</th>
<th>Safe?</th>
</tr>
</thead>
<tbody>
<tr>
<td>unique id</td>
<td>aws ec2 describe-images --imageIds ami-5731123e</td>
<td>✓</td>
</tr>
<tr>
<td>owner and name</td>
<td>aws ec2 describe-images --owners myOrg \</td>
<td>✓</td>
</tr>
<tr>
<td>just name</td>
<td>--filters &quot;Name=name,Values=ubuntu16.04-*&quot;</td>
<td>X</td>
</tr>
</tbody>
</table>

Real-world example: AWS API

```
DescribeImagesRequest request = new DescribeImagesRequest();
request.withFilters(new Filter("name", "RHEL-7.5_HVM_GA"));
request.withOwners("myOrg");
api.describeImages(request);
```

Unsafe: searches the public repository!

Unsafe without this line!

Preventing sniping: track calls

How to use:

1. Write specification on API once:

   ```
   DescribeImageResponse describeImages(
   @CalledMethods("withImageIds || withOwners")
   DescribeImageRequest request) { … }
   ```

2. Prove code correct using a type system and local inference:

   ```
   @CalledMethods(A) Object o ⇒
   ∀ a ∈ A, o.a() has definitely been called.
   ```

   ```
   @CalledMethods([{}]) Object
   @CalledMethods(["foo", "bar"]) Object
   ```

Another use: required fields in builders

```java
@Builder
public class UserIdentity {
    private final @NonNull String name;
    private final @NonNull String displayName;
    private final @NonNull ByteArray id;
}
```

Example vulnerability from https://github.com/Netflix/SimianArmy

```
public List<Image> describeImages(String... imageIds) {
    DescribeImagesRequest request = new DescribeImagesRequest();
    if (imageIds != null) {
        request.setImageIds(AsList.asList(imageIds));
    }
    DescribeImagesResult result = ec2client.describeImages(request);
    return result.getImages();
}
```

What if `imageIds` was null? Then everything in the public repository is returned! No filter is applied afterward. There was one call site in the project that explicitly passed `null`, so this project is vulnerable!

Evaluation

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>No. projects</td>
<td>548</td>
</tr>
<tr>
<td>Source LoC</td>
<td>9.2M</td>
</tr>
<tr>
<td>True positives</td>
<td>14</td>
</tr>
<tr>
<td>False positives</td>
<td>3</td>
</tr>
</tbody>
</table>

Every project contained at least one call to an image fetching API (and was therefore potentially vulnerable to sniping).

Real-world example:

```
name is @NonNull ⇒ must call name() before build()
```

Clients must call all three of these methods before build!

Preliminary user study results:

Subjects: 6 industrial developers
Task: add a new @NonNull field to a builder, and update all call sites
Results:

- 6/6 succeeded with our tool, only 3/6 without
- Those who succeeded at both 1.5x faster with our tool
- "It was easier to have the tool report issues at compile time"

Case studies:

- 5 projects: 2 Lombok, 3 AutoValue (~500k sloc)
- 563 calls verified, 1 true positive (google/gapic-generator)
- 110 annotations, 19 false positives