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:

Filters	Example	Safe?
unique id	`aws ec2 describe-images --imageIds ami-5731123e`	✓
owner and name	`aws ec2 describe-images --owners myOrg \ --filters "Name=name,Values=ubuntu16.04-*"`	✓
just name	`aws ec2 describe-images \ --filters "Name=name,Values=ubuntu16.04-*"`	X

Unsafe: searches the public repository!

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 without this line!

Preventing sniping: track calls

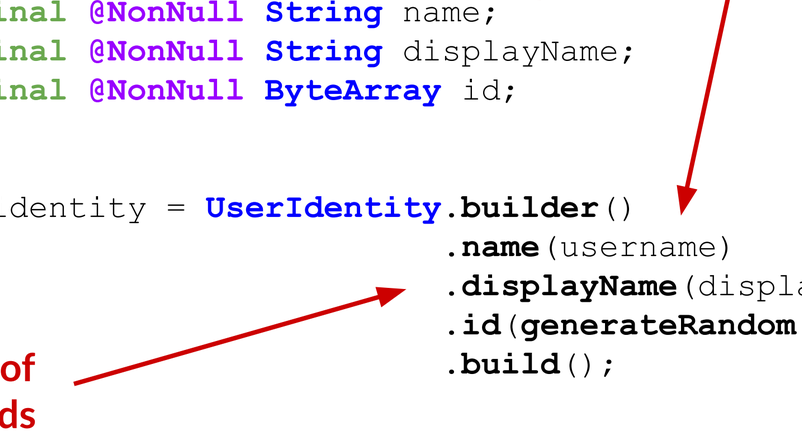
How to use:

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.



Evaluation

No. projects	548
Source LoC	9.2M
True positives	14
False positives	3

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

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

```
public List<Image> describeImages(String... imageIds) {
   DescribeImagesRequest request =
            new DescribeImagesRequest();

   if (imageIds != null) {
      request.setImageIds(Arrays.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 callsite in the project that explicitly passed `null`, so this project is vulnerable!

Another use: required fields in builders

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

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

UserIdentity identity = UserIdentity.builder()
                                    .name(username)
                                    .displayName(displayName)
                                    .id(generateRandom(32))
                                    .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