Scaling the Security Researcher to Eliminate OSS Vulnerabilities Once and for All

- Jonathan Leitschuh -
- Patrick Way -

Hello!

- Jonathan Leitschuh -

Software Engineer & Security Researcher

Dan Kaminsky Fellowship @ HUMAN Security

GitHub Star & GitHub Security Ambassador

Twitter: @JLLeitschuh

GitHub: JLLeitschuh

Hello!

- Patrick Way -

Senior Software Engineer

OpenRewrite Team @ Moderne

Twitter: @WayPatrick

GitHub: pway99

Disclaimer

Supported by
The
Dan Kaminsky Fellowship
at
HUMAN Security

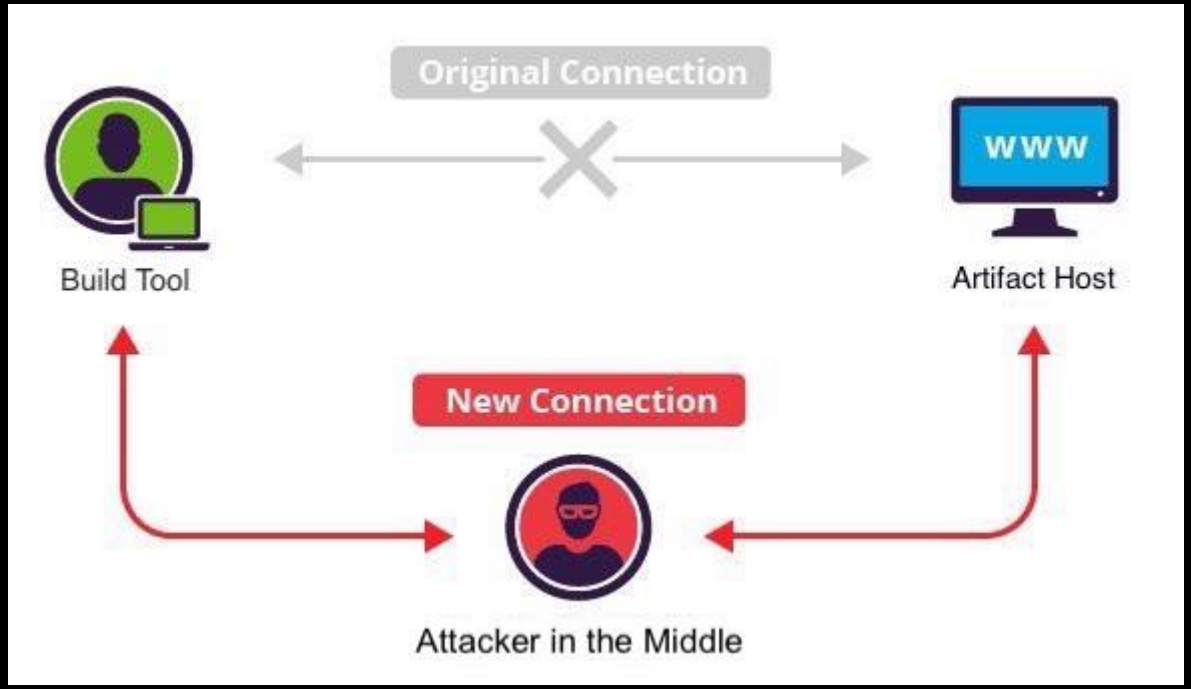
Chester Higgins/The New York Times

It Started
With a Simple Vulnerability

```
// build.gradle

maven {
  setUrl("http://dl.bintray.com/kotlin/ktor")
}
```

HTTP Download of Dependencies in the Java Ecosystem

Why is HTTPS important?



```
<!-- Compiler & Test Dependencies -->
<repositories>
  <repository>
    <id>example-id</id>
    <name>Example insecure repository</name>
    <url>http://[SOME URL HERE]</url>
  </repository>
</repositories>
```

HTTP Download of Dependencies in the Java Ecosystem

```
<!-- Artifact upload - Credentials!! -->
<distributionManagement>
  <repository>
    <id>example-id</id>
    <name>Example insecure repository</name>
    <url>http://[SOME URL HERE]</url>
  </repository>
</distributionManagement>
```

HTTP Download of Dependencies in the Java Ecosystem

This Vulnerability was Everywhere!

Who else was vulnerable?

ORACLE®

LinkedIn®

stripe

“25% of Sonatype Maven Central downloads are still using HTTP”

- Sonatype June 2019 -

How do we fix this?

Decommissioning HTTP Support

On or around January 15th, 2020

- Maven Central (Sonatype)
- JCenter (JFrog)
- Spring (Pivotal)
- Gradle Plugin Portal (Gradle)

“20% of Sonatype Maven Central Traffic is STILL using HTTP”

- Sonatype January 2020 -

You can imagine what happened...
January 15th, 2020

We stopped the bleeding

What about the other repositories?

Only the *most commonly* used repositories

- Maven Central (Sonatype)
- JCenter (JFrog)
- Spring (Pivotal)
- Gradle Plugin Portal (Gradle)

How do we fix the rest?

Bulk Pull Request Generation!

How?

CodeQL

```
import java
import semmle.code.xml.MavenPom

private class DeclaredRepository extends PomElement {
  DeclaredRepository() {
    this.getName() = "repository" or
    this.getName() = "snapshotRepository" or
    this.getName() = "pluginRepository"
  }

  string getUrl() { result = getAChild("url").(PomElement).getValue() }

  predicate isInsecureRepositoryUsage() {
    getUrl().matches("http://%") or
    getUrl().matches("ftp://%")
  }
}

from DeclaredRepository repository
where repository.isInsecureRepositoryUsage()
select repository,
  "Downloading or uploading artifacts over insecure protocol (eg. http or ftp) to/from repository " +
    repository.getUrl()
```

CodeQL scans 100Ks of OSS Projects

CodeQL

```
import java
import semmle.code.xml.MavenPom

private class DeclaredRepository extends PomElement {
  DeclaredRepository() {
    this.getName() = "repository" or
    this.getName() = "snapshotRepository" or
    this.getName() = "pluginRepository"
  }

  string getUrl() { result = getAChild("url").(PomElement).getValue() }

  predicate isInsecureRepositoryUsage() {
    getUrl().matches("http://%") or
    getUrl().matches("ftp://%")
  }
}

from DeclaredRepository repository
where repository.isInsecureRepositoryUsage()
select repository,
  "Downloading or uploading artifacts over insecure protocol (eg. http or ftp) to/from repository " +
    repository.getUrl()
```

$2,300 Bounty

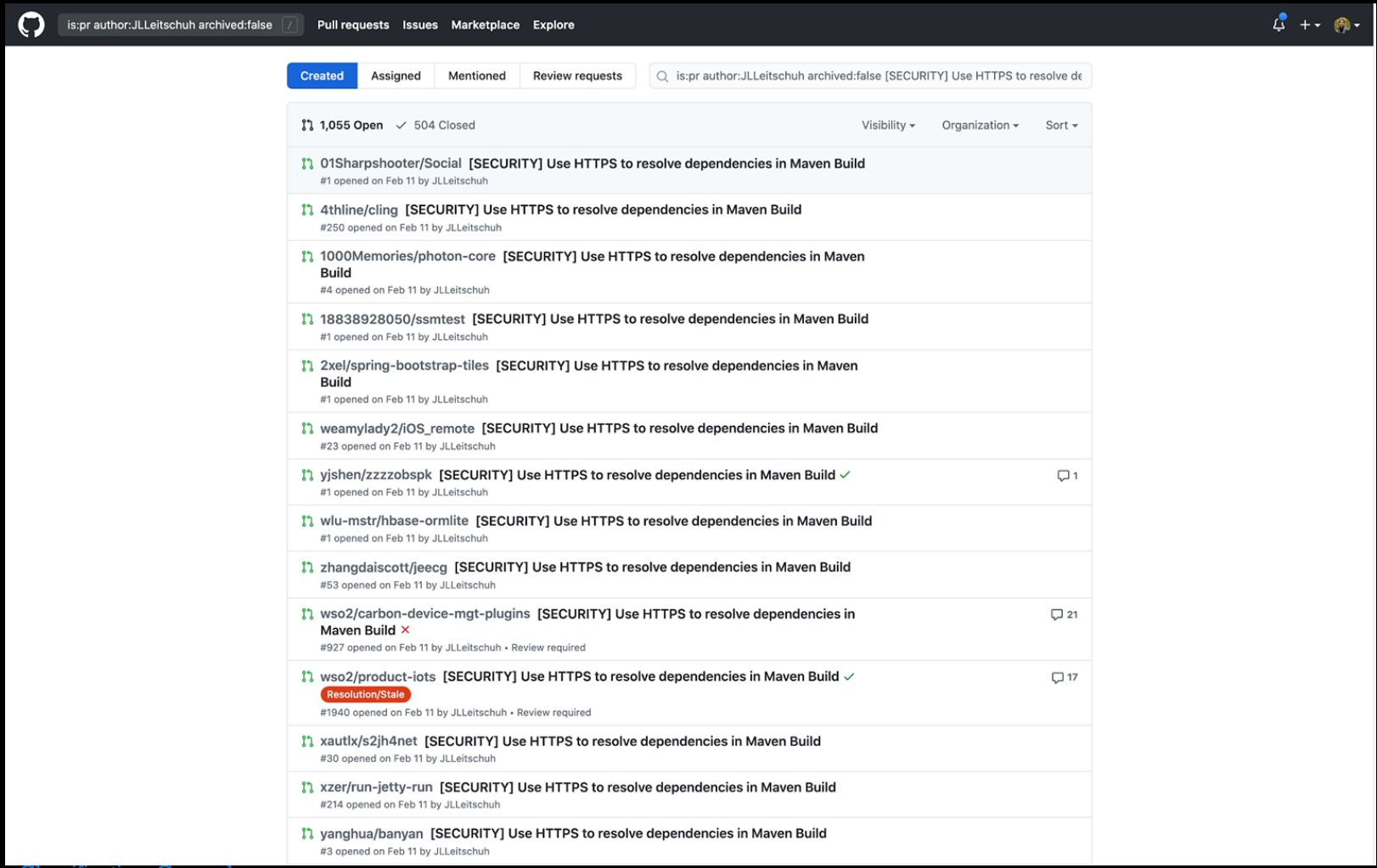
Pull Request Generator Version 1

- Python Based
- Wrapper over ‘hub’ CLI
- One Nasty Regular Expression
- Bouncing off GitHub’s rate limiter

```python
from vulnerability_fix_engine import VulnerabilityFixModule


@dataclass()
class PomVulnerabilityFixModule(VulnerabilityFixModule):
    branch_name: str = 'fix/JLL/use_https_to_resolve_dependencies'
    clone_repos_location: str = 'cloned_repos'
    data_base_dir: str = 'insecure_pom_data'
    save_point_location: str = 'save_points'
    pr_message_file_absolute_path: str = f'{str(pathlib.Path().absolute())}/PR_MESSAGE.md'
    commit_message: str = textwrap.dedent('''\
        Use HTTPS instead of HTTP to resolve dependencies

        This fixes a security vulnerability in this project where the `pom.xml`
        files were configuring Maven to resolve dependencies over HTTP instead of
        HTTPS.

        Signed-off-by: Jonathan Leitschuh <Jonathan.Leitschuh@gmail.com>
        ''')

    p_fix_regex = \
        re.compile(
            r'(?:(?<=<repository>)|(?<=<pluginRepository>)|(?<=<snapshotRepository>))((?:(?!repository>).)*)(<url>\s*)http://(\S*)(\s*</url>)',
            re.IGNORECASE + re.MULTILINE + re.DOTALL
        )
    replacement = r'\1\2https://\3\4'

    async def do_fix_vulnerable_file(self, project_name: str, file: str, expected_fix_count: int) -> int:
        async with aiofiles.open(file, newline='') as vulnerableFile:
            contents: str = await vulnerableFile.read()

        new_contents, count = self.p_fix_regex.subn(self.replacement, contents)
        if count != expected_fix_count:
            logging.warning(
                'Fix for `%s` did match expected fix count: (expected: %d, actual: %d)',
                project_name,
                expected_fix_count,
                count
            )

        async with aiofiles.open(file, 'w', newline='') as vulnerableFile:
            await vulnerableFile.write(new_contents)
        return count
```

```
p_fix_regex = \
    re.compile(
        r'(?:(?<=<repository>)|(?<=<pluginRepository>)|(?<=<snapshotRepository>))((?:(?!repository>).)*)(<url>\s*)http://(\S*)(\s*</url>)',
        re.IGNORECASE + re.MULTILINE + re.DOTALL
    )
replacement = r'\1\2https://\3\4'
```

It worked!



HTTP Download of Dependencies

1,596

Pull Requests

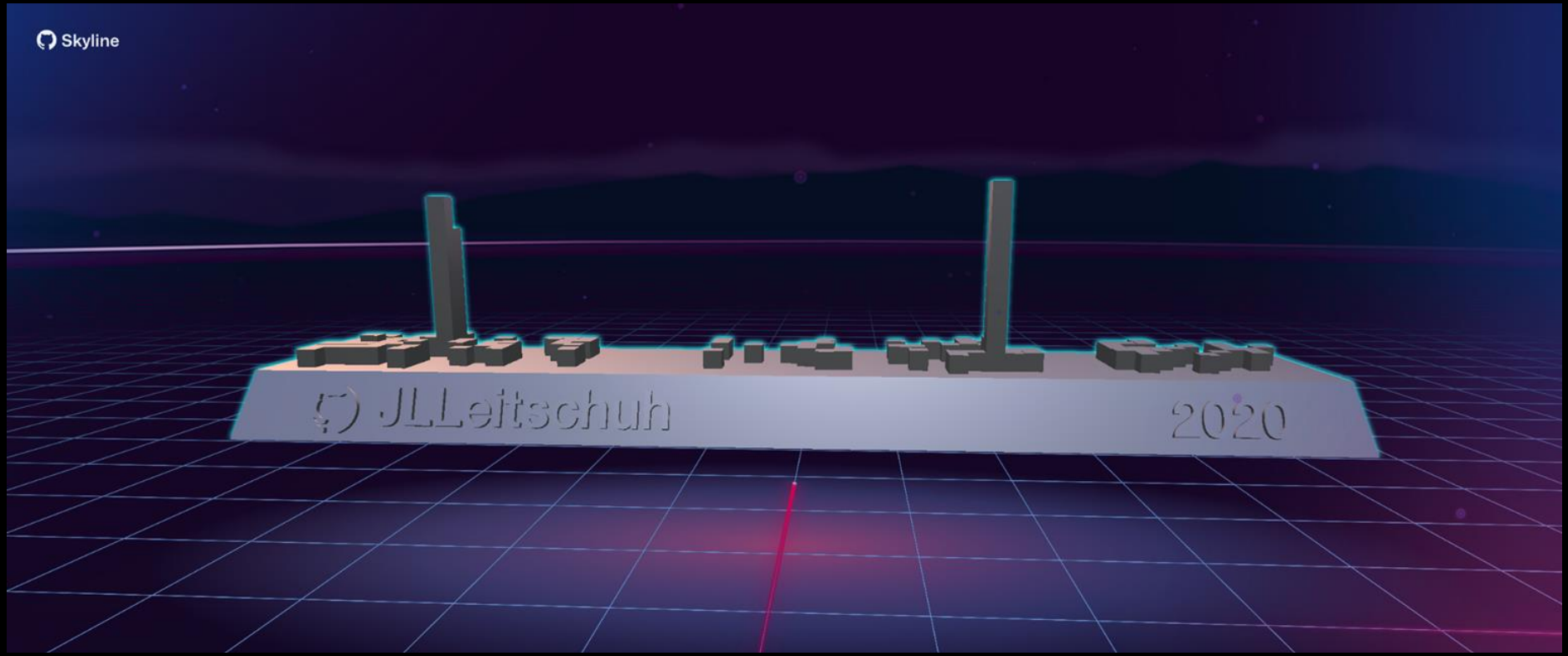
~40%

Merged or Accepted

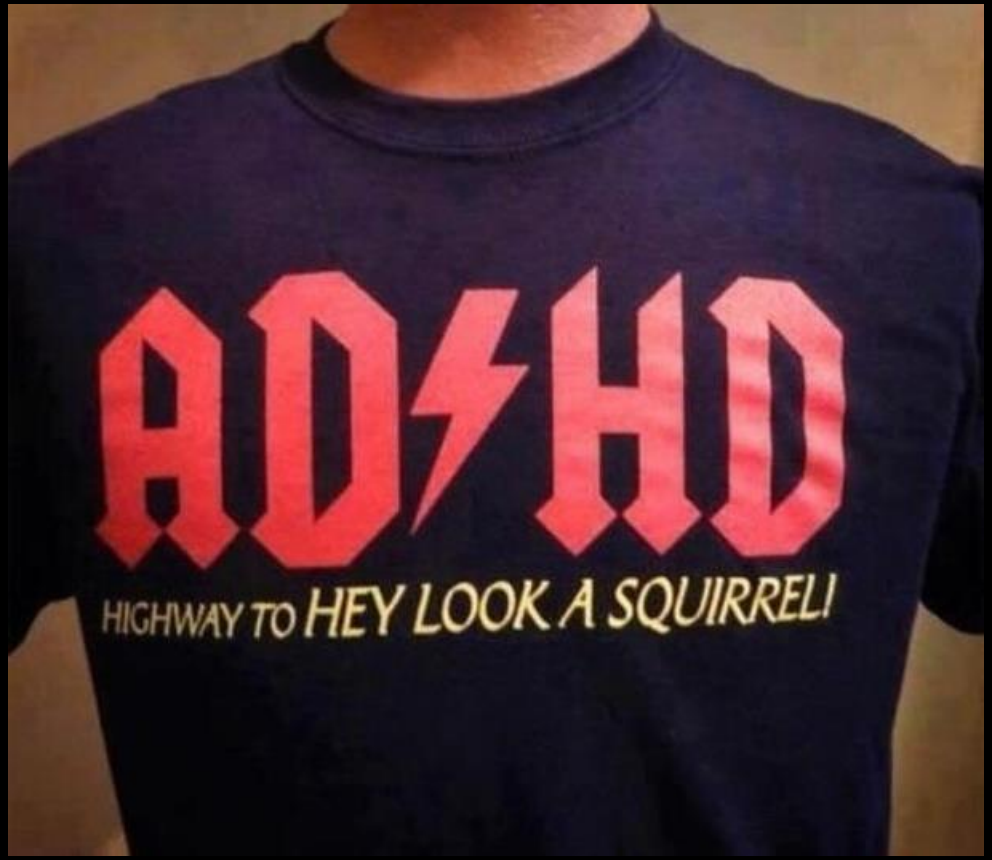
$4,000

Thanks to the GitHub Security Lab!

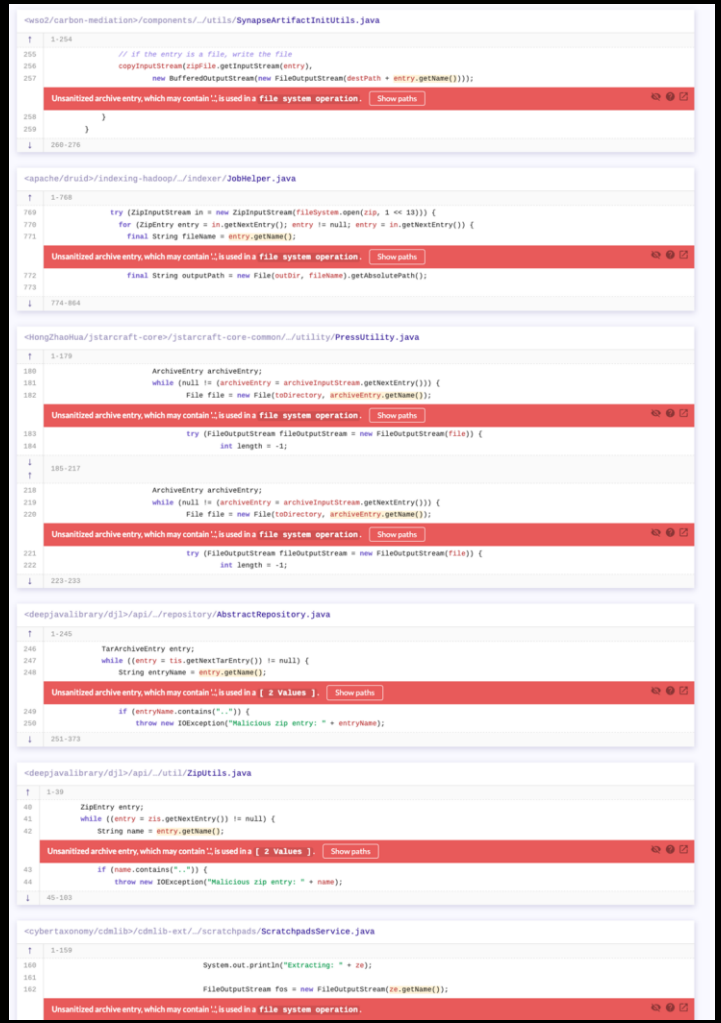
I got hooked on
Bulk Pull Request Generation



I have a Problem



I was finding too many security vulnerabilities!



I was finding too many security vulnerabilities!

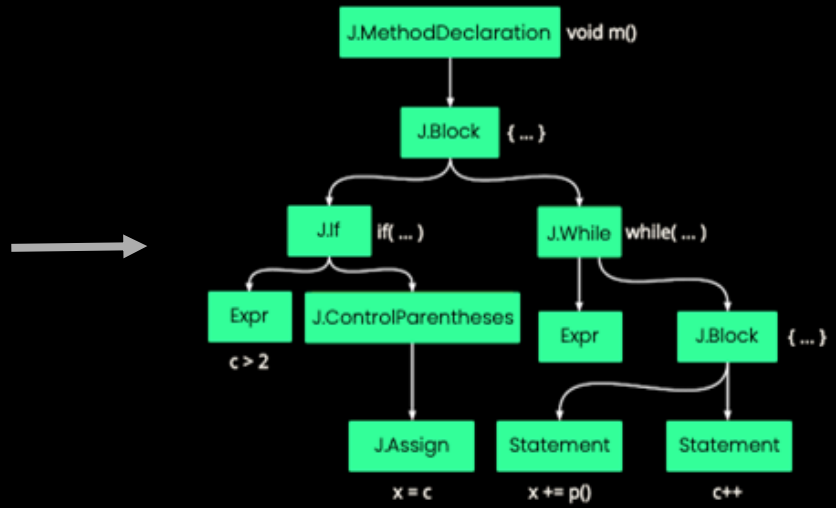
I was finding too many security vulnerabilities!

I needed automation!

Automated Accurate Transformations at a Massive Scale

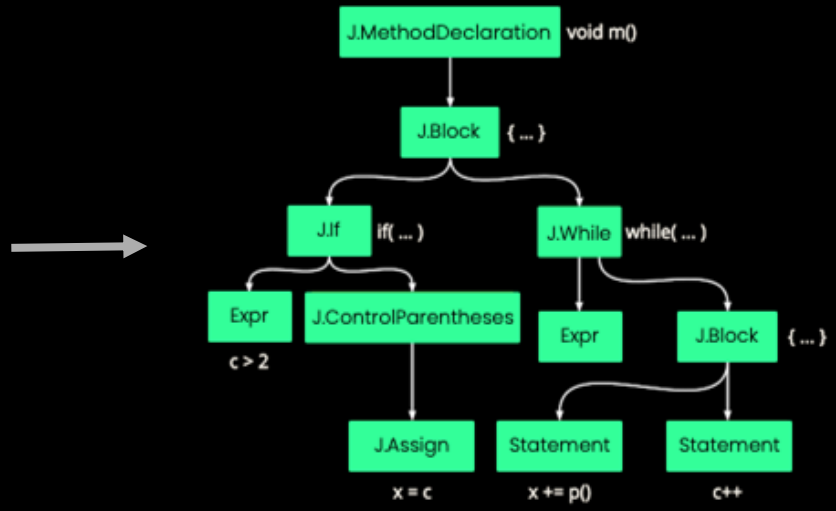
Abstract Syntax Tree (AST)

```java
/** myMethod */
void m(){
    if (c > 2) {
        // c is more than 2
        x = c;
    }
    while(c < 10) { // increment x
        x += p();
        c++;
    }
}
```



Abstract Syntax Tree (AST)

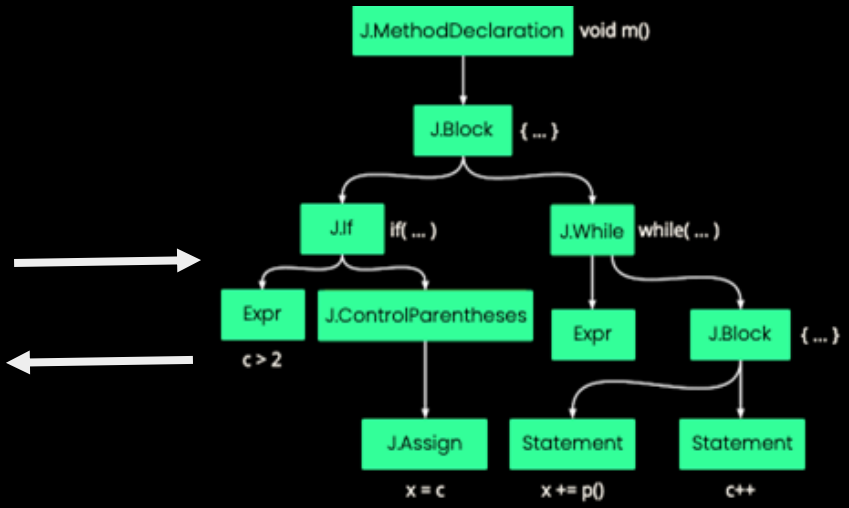
```java
/** myMethod */
void m(){
    if (c > 2) {
        // c is more than 2
        x = c;
    }
    while(c < 10) { // increment x
        x += p();
        c++;
    }
}
```




```
void m(){if(c>2){x=c;}while(c<10){x+=p();c++;}}
```

Format Preserving AST

```java
/** myMethod */
void m(){
    if (c > 2) {
        // c is more than 2
        x = c;
    }
    while(c < 10) { // increment x
        x += p();
        c++;
    }
}
```



Whitespace and comments are preserved

Generated code matches the Surrounding Formatting

Spaces

```
String name = entry.getName();
Path path = dir.resolve(name);
if (!path.normalize().startsWith(dir)) {
    throw new RuntimeException("Bad zip entry");
}
OutputStream os = Files.newOutputStream(path);
```

Tabs

```
Path path = dir.resolve(name);
if (!path.normalize().startsWith(dir)) {
	throw new RuntimeException("Bad zip entry");
}
OutputStream os = Files.newOutputStream(path);
```

Braces on new line

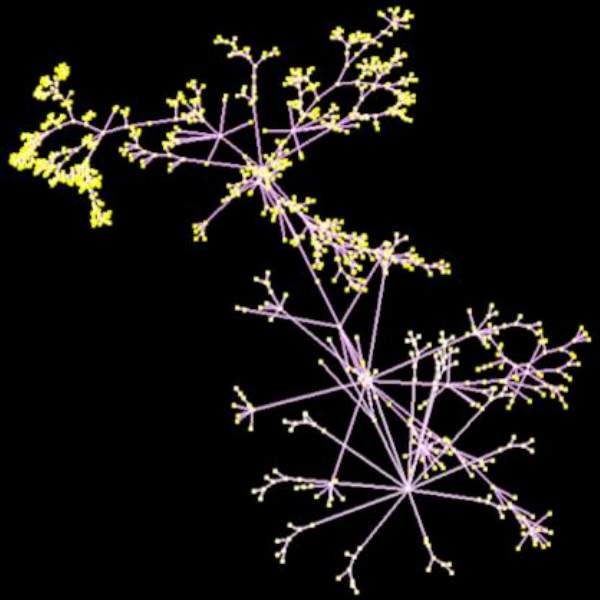
```
String name = entry.getName();
Path path = dir.resolve(name);
if (!path.normalize().startsWith(dir))
{
    throw new RuntimeException("Bad zip entry");
}
OutputStream os = Files.newOutputStream(path);
```

Accurate Transformations Require Fully Type-attributed ASTs

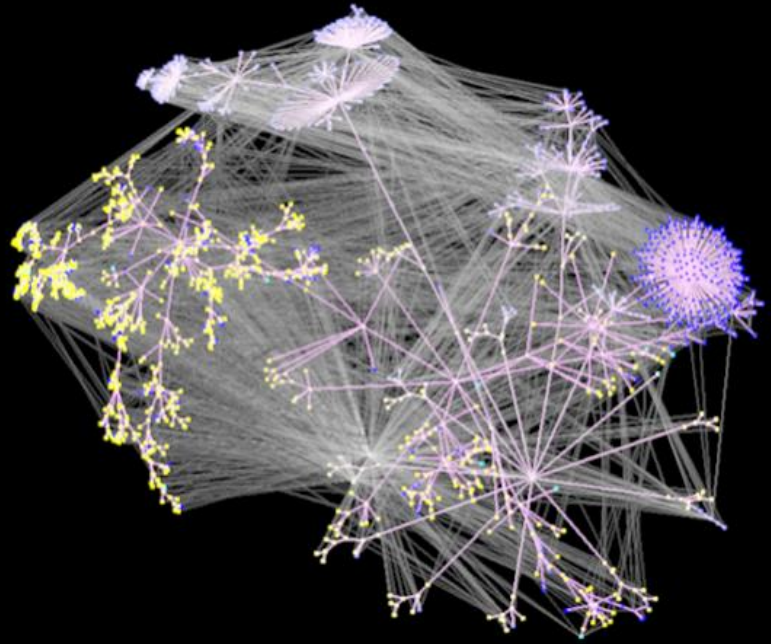
```
log.info("...");
```

Is that log4j, slf4j, LogBack?

The OpenRewrite AST is both Syntactically and Semantically aware.



Syntax alone



With type attribution and formatting

Even simple code produces complex AST

```
if (!path.normalize().startsWith(dir)) {
    throw new RuntimeException("Bad zip entry");
}
```

```
final JavaTemplate noZipSlipPathStartsWithPathTemplate =
JavaTemplate.builder(this::getCursor, code: "" +
"if (!#{any(java.nio.file.Path)}.normalize()" +
        ".startsWith(#{any(java.nio.file.Path)})) {\n" +
"    throw new RuntimeException(\"Bad zip entry\");\n" +
"}").build();
```

```
final JavaTemplate noZipSlipPathStartsWithPathTemplate =
JavaTemplate.builder(this::getCursor, code: "" +
"if (!#{any(java.nio.file.Path)}.normalize()" +
        ".startsWith(#{any(java.nio.file.Path)})) {\n" +
"    throw new RuntimeException(\"Bad zip entry\");\n" +
"}").build();
```

```
return b.withTemplate(
        noZipSlipPathStartsWithPathTemplate,
        zipSlipSimpleInjectGuardInfo.statement.getCoordinates().after(),
        zipSlipSimpleInjectGuardInfo.zipEntry,
        zipSlipSimpleInjectGuardInfo.parentDir
);
```

```java
public class MyZipHelper {
    public void m1(ZipEntry entry, Path dir) throws Exception {
        String name = entry.getName();
        Path path = dir.resolve(name);
        OutputStream os = Files.newOutputStream(path);
    }
}
```



```java
public class MyZipHelper {
    public void m1(ZipEntry entry, Path dir) throws Exception {
        String name = entry.getName();
        Path path = dir.resolve(name);
        if (!path.normalize().startsWith(dir)) {
            throw new RuntimeException("Bad zip entry");
        }
        OutputStream os = Files.newOutputStream(path);
    }
}
```

What is possible now?

What other vulnerabilities can we fix?

Three Vulnerabilities

1. Temporary Directory Hijacking
2. Partial Path Traversal
3. Zip Slip

Vulnerability #1

Temporary Directory Hijacking

Temporary Directory on Unix-Like Systems is Shared between All Users

Temporary Directory Hijacking - Vulnerable

```
File f = File.createTempFile(
        "prefix",
        "suffix"
);
f.delete();
f.mkdir();
```

Temporary Directory Hijacking - Vulnerable

```
File f = File.createTempFile(
        "prefix",
        "suffix"
);
f.delete();
f.mkdir();
```

Temporary Directory Hijacking - Vulnerable

```
File f = File.createTempFile(
        "prefix",
        "suffix"
);
f.delete();
// 🏁 Race condition
f.mkdir(); // Returns `false`
```

Temporary Directory Hijacking - Imperfect Fix

```
File f = File.createTempFile(
        "prefix",
        "suffix"
);
f.delete();
if(!f.mkdir())
    throw new IOException("Error");
```

Temporary Directory Hijacking - Fix

```
// Since Java 1.7
File f =
    Files
        .createTempDirectory("prefix")
        .toFile();
```

Temporary Directory Hijacking - CVEs

- CVE-2022-27772 - Spring Boot
- CVE-2021-20202 - Keycloak
- CVE-2021-21331 - DataDog API
- CVE-2020-27216 - Eclipse Jetty
- CVE-2020-17521 - Apache Groovy
- CVE-2020-17534 - Apache netbeans-html4j

Temporary Directory Hijacking

Pull Request Statistics

Temporary Directory Hijacking

64 Pull Requests!

Temporary Directory Hijacking - Pull Requests

Temporary Directory Hijacking - Putting it all together

src/main/java/org/jenkinsci/backend/jpicreate/WebAppMain.java

```
@@ -10,6 +10,7 @@ org.openrewrite.java.security.UseFilesCreateTempDirectory
10  10    import javax.sound.midi.SysexMessage;
11  11    import java.io.File;
12  12    import java.io.IOException;
    13  + import java.nio.file.Files;
13  14
14  15    /**
15  16     *
@@ -41,9 +42,7 @@
41  42            FileUtils.copyURLToFile(
42  43                    getClass().getClassLoader().getResource("maven.zip"),
43  44                    zip);
44      -         File bin = File.createTempFile("maven","bin");
45      -         bin.delete();
46      -         bin.mkdirs();
    45  +         File bin = Files.createTempDirectory("maven" + "bin").toFile();
47  46
48  47            Process unzip = new ProcessBuilder("unzip", zip.getAbsolutePath())
49  48                    .directory(bin).redirectErrorStream(true).start();
```

Temporary Directory Hijacking - Putting it all together

src/test/java/com/google/jenkins/plugins/credentials/oauth/JsonServiceAccountConfigTestUtil.java

```
@@ -22,6 +22,7 @@ org.openrewrite.java.security.UseFilesCreateTempDirectory
22  22    import java.io.IOException;
23  23    import java.io.StringWriter;
24  24    import java.nio.charset.Charset;
    25  +  import java.nio.file.Files;
25  26    import java.security.KeyPair;
26  27    import java.security.KeyPairGenerator;
27  28    import java.security.NoSuchAlgorithmException;
@@ -64,13 +65,7 @@
64  65
65  66      private static File getTempFolder() throws IOException {
66  67        if (tempFolder == null) {
67      -       tempFolder = File.createTempFile("temp", Long.toString(System.nanoTime()));
68      -       if (!tempFolder.delete()) {
69      -         throw new IOException("Could not delete temp file: " + tempFolder.getAbsolutePath());
70      -       }
71      -       if (!tempFolder.mkdir()) {
72      -         throw new IOException("Could not create temp directory: " + tempFolder.getAbsolutePath());
73      -       }
    68  +       tempFolder = Files.createTempDirectory("temp" + Long.toString(System.nanoTime())).toFile();
74  69          tempFolder.deleteOnExit();
75  70        }
76  71        return tempFolder;
```

Vulnerability #2

Partial Path Traversal

Partial Path Traversal

```
"/user/sam"
```

Partial Path Traversal

```
"/user/sam"
```

```
"/user/samantha"
```

Partial Path Traversal

Allows an attacker access to a sibling directory with the same prefix

Partial Path Traversal

```
"/user/sam"
```

Allows an attacker access to a sibling directory with the same prefix

Partial Path Traversal

```
"/user/sam"
```

Allows an attacker access to a sibling directory with the same prefix

```
"/user/samantha"
```

Partial Path Traversal

```
"/user/sam"
```

Allows an attacker access to a sibling directory with the same prefix

```
"/user/samantha"
```

Partial Path Traversal - Vulnerability

```
File dir = new File(
    parent, userControlled()
);

if (!dir.getCanonicalPath()
        .startsWith(parent.getCanonicalPath())) {
    throw new IOException(
            "Detected path traversal attack!"
    );
}
```

```
new File("/user/sam/")
```

```
new File("/user/sam/")
```

File.getCanonicalPath()

```
new File("/user/sam/")
```

File.getCanonicalPath()

```
"/user/sam"
```

```
new File("/user/sam/")
```

File.getCanonicalPath()

```
"/user/sam"
```

Partial Path Traversal - Vulnerability

```
File dir = new File(
    parent, userControlled()
);

if (!dir.getCanonicalPath()
        .startsWith(parent.getCanonicalPath())) {
    throw new IOException(
            "Detected path traversal attack!"
    );
}
```

Partial Path Traversal - Vulnerability

```
File dir = new File(
    "/user/sam/", userControlled()
);

if (!dir.getCanonicalPath()
        .startsWith("/user/sam")) {
    ...



}
```

Partial Path Traversal - Vulnerability

```
File dir = new File(
    "/user/sam/", "../samantha/baz"
);

if (!dir.getCanonicalPath()
        .startsWith("/user/sam")) {
    ...



}
```

Partial Path Traversal - Vulnerability

```
File dir = new File(
    "/user/sam/", "../samantha/baz"
);

if (!"/user/samantha/baz"
        .startsWith("/user/sam")) {
    ...



}
```

Partial Path Traversal - Vulnerability

```
File dir = new File(
    "/user/sam/", "../samantha/baz"
);

if (!"/user/samantha/baz"
        .startsWith("/user/sam")) {
    throw new IOException(
            "Detected path traversal attack!"
    );
}
```

Partial Path Traversal Fix!

Partial Path Traversal - Vulnerability

```
File dir = new File(
    parent, userControlled()
);
if (!dir.getCanonicalPath()
        .startsWith(parent.getCanonicalPath())) {
    throw new IOException(
            "Detected path traversal attack!"
    );
}
```

Partial Path Traversal - Vulnerability

```
if (!dir.getCanonicalPath()
        .startsWith(parent.getCanonicalPath())) {
    ...


}
```

Partial Path Traversal - Fix #1

```
if (!dir.getCanonicalPath()
        .startsWith(parent.getCanonicalPath() +
                    File.separatorChar)) {
    ...

}
```

Partial Path Traversal - Fix #2

```
if (!dir.getCanonicalFile()
        .toPath().startsWith(
            parent.getCanonicalFile().toPath())) {
    ...

}
```

Partial Path Traversal - Fix #2 - Better

```
if (!dir.getCanonicalFile()
        .toPath().startsWith(
            parent.getCanonicalFile().toPath())) {
    ...

}
```

How do we find this vulnerability?

Partial Path Traversal - Vulnerability

```
File dir = new File(
    parent, userControlled()
);
if (!dir.getCanonicalPath()
        .startsWith(parent.getCanonicalPath())) {
    throw new IOException(
            "Detected path traversal attack!"
    );
}
```

Partial Path Traversal - Vulnerability

```
if (!dir.getCanonicalPath()
        .startsWith(parent.getCanonicalPath())) {
    ...


}
```

Partial Path Traversal - Vulnerability

```
if (!dir.getCanonicalPath()
        .startsWith(parent.getCanonicalPath())) {
    ...


}
```

Partial Path Traversal - Safe

```
if (!dir.getCanonicalPath()
        .startsWith(parent.getCanonicalPath() +
                    File.separatorChar)) {
    ...

}
```

It can’t be that easy, can it?

Partial Path Traversal - Vulnerability

```
if (!dir.getCanonicalPath()
        .startsWith(parent.getCanonicalPath())) {
    ...


}
```

Partial Path Traversal - Vulnerability

```
String dirCanonical = dir.getCanonicalPath();


if (!dirCanonical
        .startsWith(parent.getCanonicalPath())) {
    ...


}
```

Partial Path Traversal - Vulnerability

```
String dirCanonical = dir.getCanonicalPath();
String pCanonical = parent.getCanonicalPath();

if (!dirCanonical
        .startsWith(pCanonical)) {
    ...
}
```

Partial Path Traversal - Vulnerability

```
String dirCanonical = dir.getCanonicalPath();
String pCanonical = parent.getCanonicalPath() +
                                File.separatorChar;
if (!dirCanonical
        .startsWith(pCanonical)) {
   ...
}
```

We need Data Flow Analysis

Partial Path Traversal - DataFlow

```
String dirCanonical = dir.getCanonicalPath();
String pCanonical = parent.getCanonicalPath() +
                                File.separatorChar;

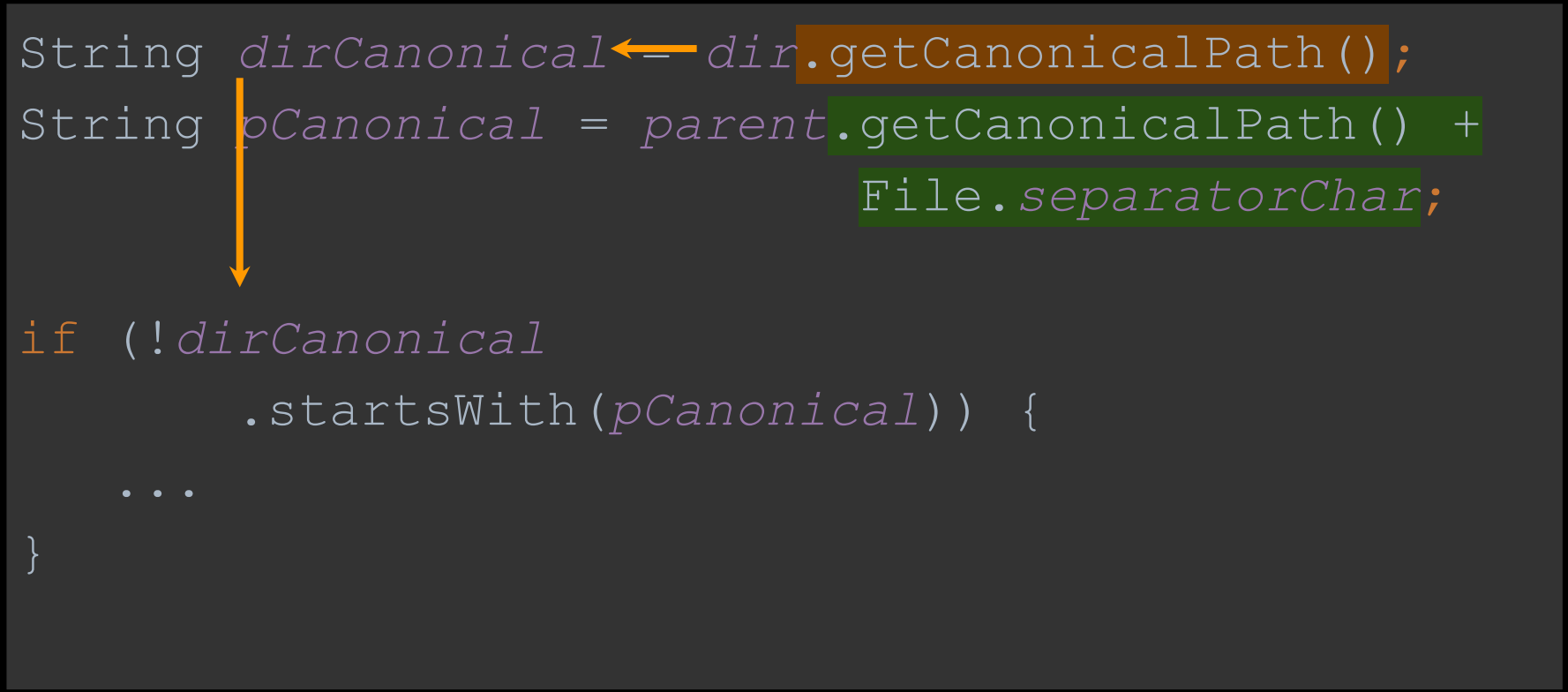
if (!dirCanonical
        .startsWith(pCanonical)) {
    ...
}
```

Partial Path Traversal - Data Flow

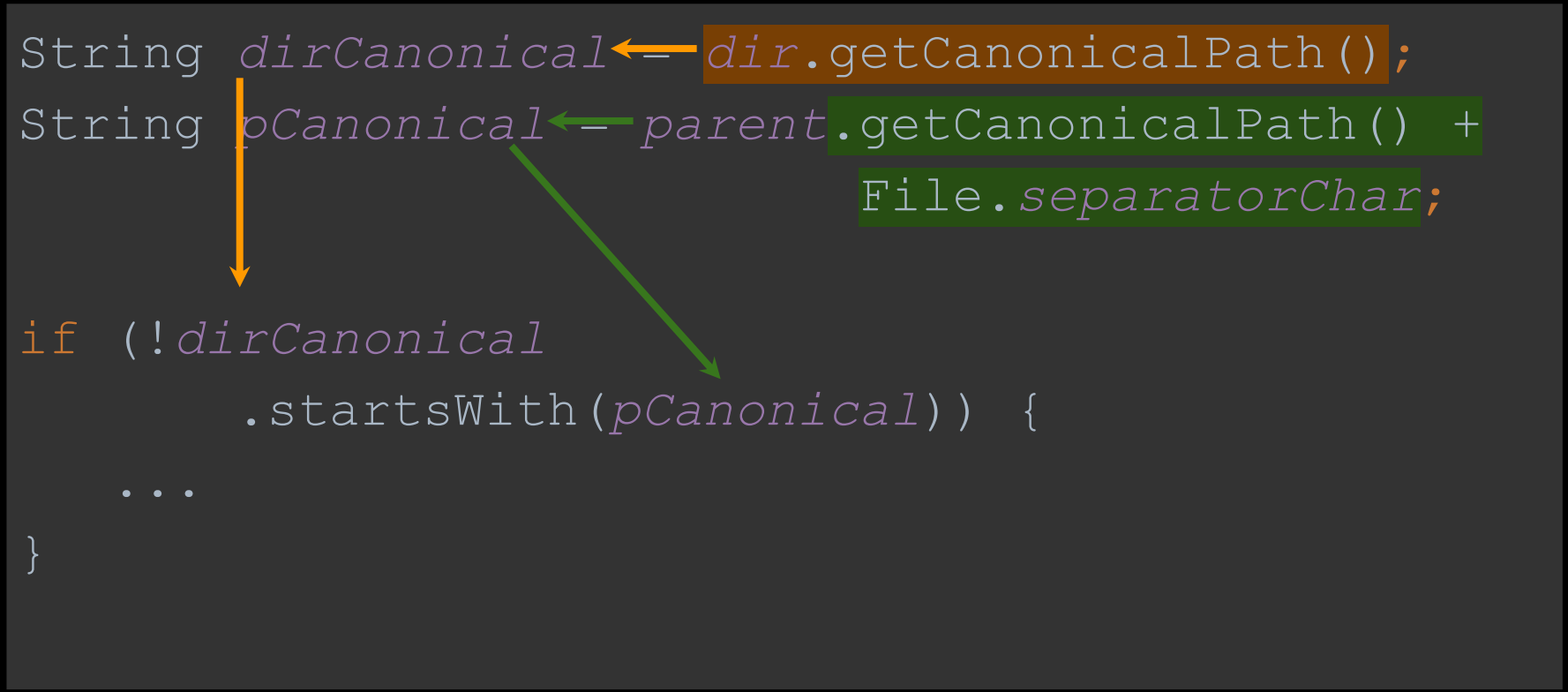
```
String dirCanonical = dir.getCanonicalPath();
String pCanonical = parent.getCanonicalPath() +
                                File.separatorChar;

if (!dirCanonical
        .startsWith(pCanonical)) {
    ...
}
```

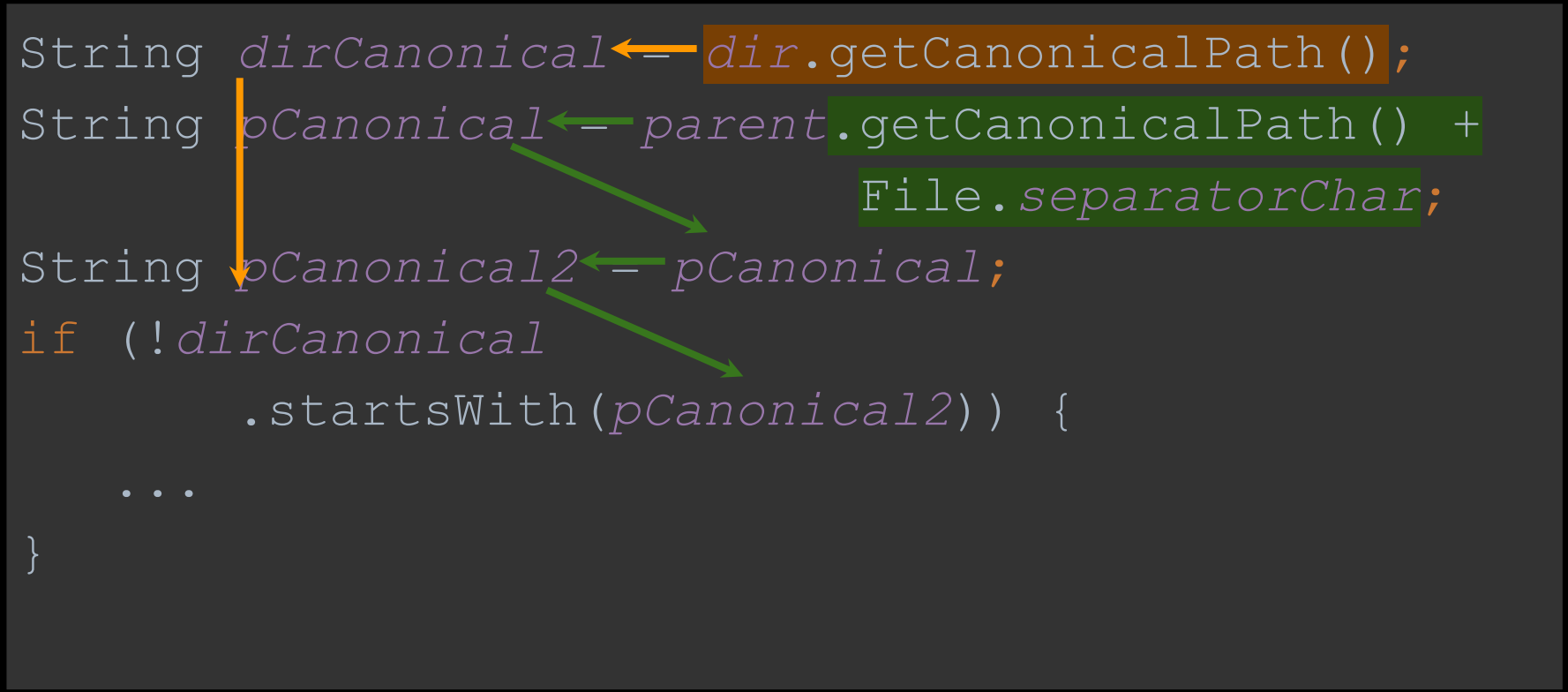
Partial Path Traversal - Data Flow



Partial Path Traversal - Data Flow



Partial Path Traversal - Data Flow



Data Flow

Uncovers hard to find Vulnerabilities

and prevents

False Positives

Data Flow Analysis

```
class GetCanonicalPathToStartsWithLocalFlow extends LocalFlowSpec<J.MethodInvocation, Expression> {

   @Override
   public boolean isSource(J.MethodInvocation methodInvocation, Cursor cursor) {
       return new MethodMatcher("java.io.File getCanonicalPath()")
               .matches(methodInvocation);
   }

   @Override
   public boolean isSink(Expression expression, Cursor cursor) {
       return InvocationMatcher
               .fromMethodMatcher(
                       new MethodMatcher(
                               "java.lang.String startsWith(java.lang.String)"
                       )
               )
               .advanced()
               .isSelect(cursor);
   }
}
```

Partial Path Traversal - Putting it all together

Example Case: AWS Java SDK CVE-2022-31159

aws-sdk-java/aws-java-sdk-s3/src/main/java/com/amazonaws/services/s3/transfer/TransferManager.java
Lines 1513 to 1519 in 5be0807

```
1513     private boolean leavesRoot(File localBaseDirectory, String key) {
1514         try {
1515             return !new File(localBaseDirectory, key).getCanonicalPath().startsWith(localBaseDirectory.getCanonicalPath());
1516         } catch (IOException e) {
1517             throw new RuntimeException("Unable to canonicalize paths",  e);
1518         }
1519     }
```

aws-sdk-java/aws-java-sdk-s3/src/main/java/com/amazonaws/services/s3/transfer/TransferManager.java
Lines 1420 to 1423 in ae88c8a

```
1420      if ( leavesRoot(destinationDirectory, s.getKey()) ) {
1421          throw new RuntimeException("Cannot download key " + s.getKey() +
1422              ", its relative path resolves outside the parent directory.");
1423      }
```

Vulnerability Disclosure Drama!

Aside: Email with AWS Security Team

AWS: We’d like to award you a bug bounty, however you’d need to sign an NDA.

Aside: Email with AWS Security Team

AWS: We'd like to award you a bug bounty, however you'd need to sign an NDA.

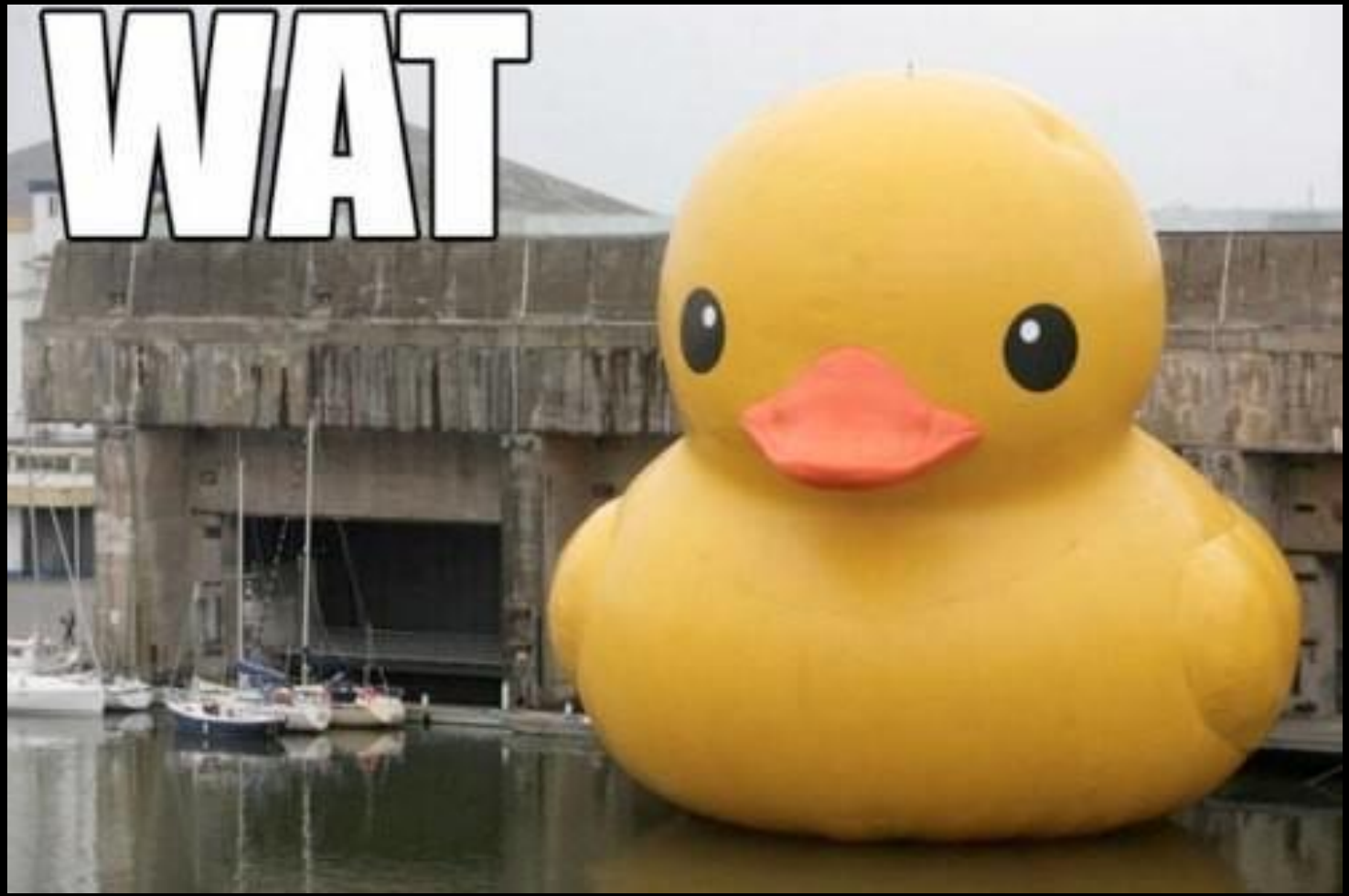
Jonathan: I don't normally agree to NDA's. Can I read it first before potentially agreeing?

Aside: Email with AWS Security Team

AWS: We'd like to award you a bug bounty, however you'd need to sign an NDA.

Jonathan: I don't normally agree to NDA's. Can I read it first before potentially agreeing?

AWS: We're unable to share the bug bounty program NDA since it and other contract documents are considered sensitive by the legal team.



Vulnerability #3

Zip Slip

Zip Slip

Path Traversal Vulnerability while Unpacking Zip File Entries

Zip Slip

```
void zipSlip(File destination, ZipFile zip) {
    Enumeration<? extends ZipEntry> entries = zip.entries();
    while (entries.hasMoreElements()) {
        ZipEntry e = entries.nextElement();
        File f = new File(destination, e.getName());
        IOUtils.copy(
                zip.getInputStream(e),
                new FileOutputStream(f)
        );
    }
}
```

Zip Slip

```
ZipEntry e = entries.nextElement();
File f = new File(destination, e.getName());
IOUtils.copy(
        zip.getInputStream(e),
        new FileOutputStream(f)
);
```

Zip Slip is Complicated

Zip Slip

```
ZipEntry e = ...
File f = new File(destination, e.getName());



IOUtils.copy(
        zip.getInputStream(e),
        new FileOutputStream(f)
);
```

Zip Slip

```
ZipEntry e = ...
File f = new File(destination, e.getName());
if (!f.toPath().startsWith(destination.toPath())) {
    throw new IOException("Bad Zip Entry!");
}
IOUtils.copy(
        zip.getInputStream(e),
        new FileOutputStream(f)
);
```

The Problem with Zip Slip

Zip Slip

```
ZipEntry e = ...
File f = new File(destination, e.getName());
if (!f.toPath().startsWith(destination.toPath())) {
    throw new IOException("Bad Zip Entry!");
}
IOUtils.copy(
        zip.getInputStream(e),
        new FileOutputStream(f)
);
```

Zip Slip

```
ZipEntry e = ...
File f = new File(destination, e.getName());
if (f.toPath().startsWith(destination.toPath())) {
    IOUtils.copy(
            zip.getInputStream(e),
            new FileOutputStream(f)
    );
}
```

Control Flow Analysis

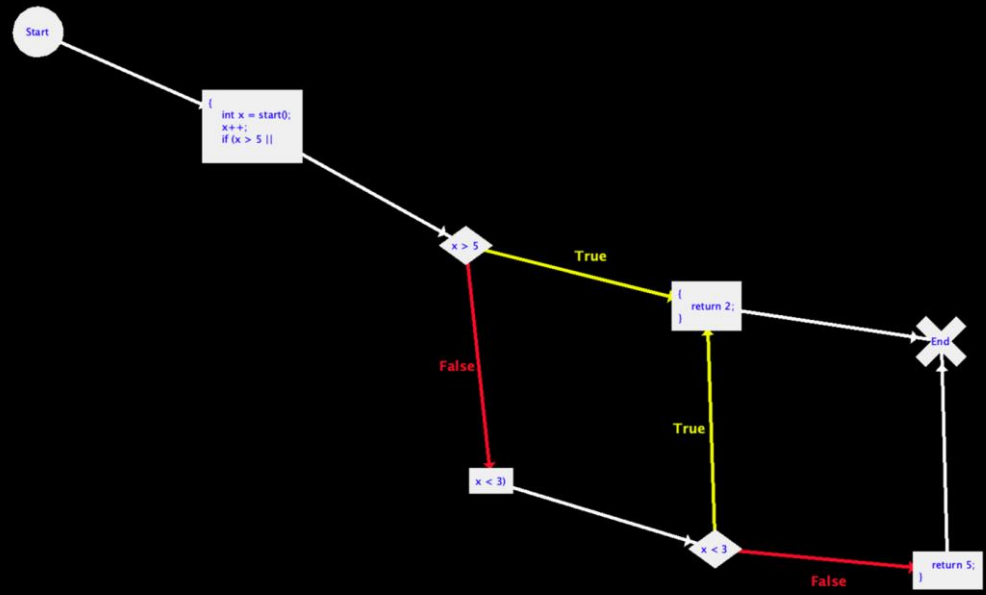
Control Flow Analysis

```
File f = new File(destination, e.getName());
IOUtils.copy(
        zip.getInputStream(e),
        new FileOutputStream(f)
);
```

```
File f = new File(destination, e.getName());
if
(!f.toPath().startsWith(destination.toPath())){
    throw new IOException("Bad Zip Entry!");
}
IOUtils.copy(
        zip.getInputStream(e),
        new FileOutputStream(f)
);
```

Control Flow - OpenRewrite

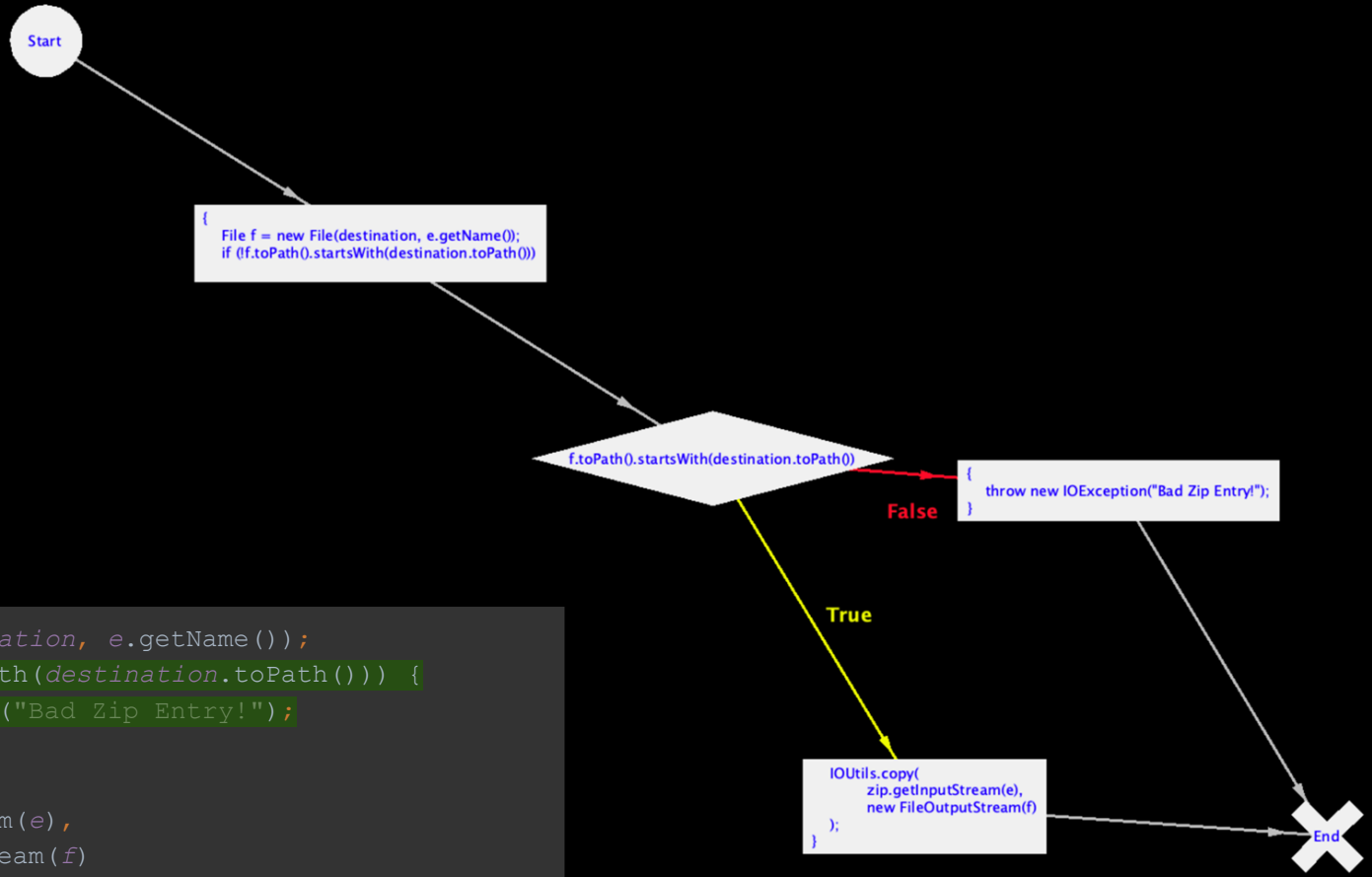
```
abstract class Test {
    abstract int start();
    int test() {
        int x = start();
        x++;
        if (x > 5 || x < 3) {
            return 2;
        }
        return 5;
    }
}
```



Zip Slip

```
ZipEntry e = ...
File f = new File(destination, e.getName());
if (!f.toPath().startsWith(destination.toPath())) {
    throw new IOException("Bad Zip Entry!");
}
IOUtils.copy(
        zip.getInputStream(e),
        new FileOutputStream(f)
);
```

Zip Slip




```
File f = new File(destination, e.getName());
if (!f.toPath().startsWith(destination.toPath())) {
   throw new IOException("Bad Zip Entry!");
}
IOUtils.copy(
       zip.getInputStream(e),
       new FileOutputStream(f)
);
```

Zip Slip - Putting it all together

Zip Slip - Putting it all together

jbake-core/src/main/java/org/jbake/app/ZipUtil.java

```
@@ -28,7 +28,10 @@ org.openrewrite.java.security.ZipSlip
28  28          byte[] buffer = new byte[1024];
29  29
30  30          while ((entry = zis.getNextEntry()) != null) {
31      -           File outputFile = new File(outputFolder.getCanonicalPath() + File.separatorChar + entry.getName());
    31  +           File outputFile = new File(outputFolder.getCanonicalPath(), entry.getName());
    32  +           if (!outputFile.toPath().normalize().startsWith(outputFolder.getCanonicalPath())) {
    33  +               throw new RuntimeException("Bad zip entry");
    34  +           }
32  35              File outputParent = new File(outputFile.getParent());
33  36              outputParent.mkdirs();
```

Pull Request Generation!

Problems with Pull Request Generation

How fast can we generate Pull Requests?

Pull Request Generation Steps

File IO Git Operation GitHub API

Pull Request Generation Steps

1. Checkout (ie. Download) code Repository

File IO Git Operation GitHub API

Pull Request Generation Steps

1. Checkout (ie. Download) code Repository
2. Branch, Apply Diff, & Commit

File IO Git Operation GitHub API

Pull Request Generation Steps

1. Checkout (ie. Download) code Repository
2. Branch, Apply Diff, & Commit
3. Fork Repository on GitHub

File IO Git Operation GitHub API

Pull Request Generation Steps

1. Checkout (ie. Download) code Repository
2. Branch, Apply Diff, & Commit
3. Fork Repository on GitHub
4. Rename Repository on GitHub

File IO Git Operation GitHub API

Pull Request Generation Steps

1. Checkout (ie. Download) code Repository
2. Branch, Apply Diff, & Commit
3. Fork Repository on GitHub
4. Rename Repository on GitHub
5. Push changes

File IO Git Operation GitHub API

Pull Request Generation Steps

1. Checkout (ie. Download) code Repository
2. Branch, Apply Diff, & Commit
3. Fork Repository on GitHub
4. Rename Repository on GitHub
5. Push changes
6. Create Pull Request on GitHub

File IO | Git Operation | GitHub API

Pull Request Generation Steps

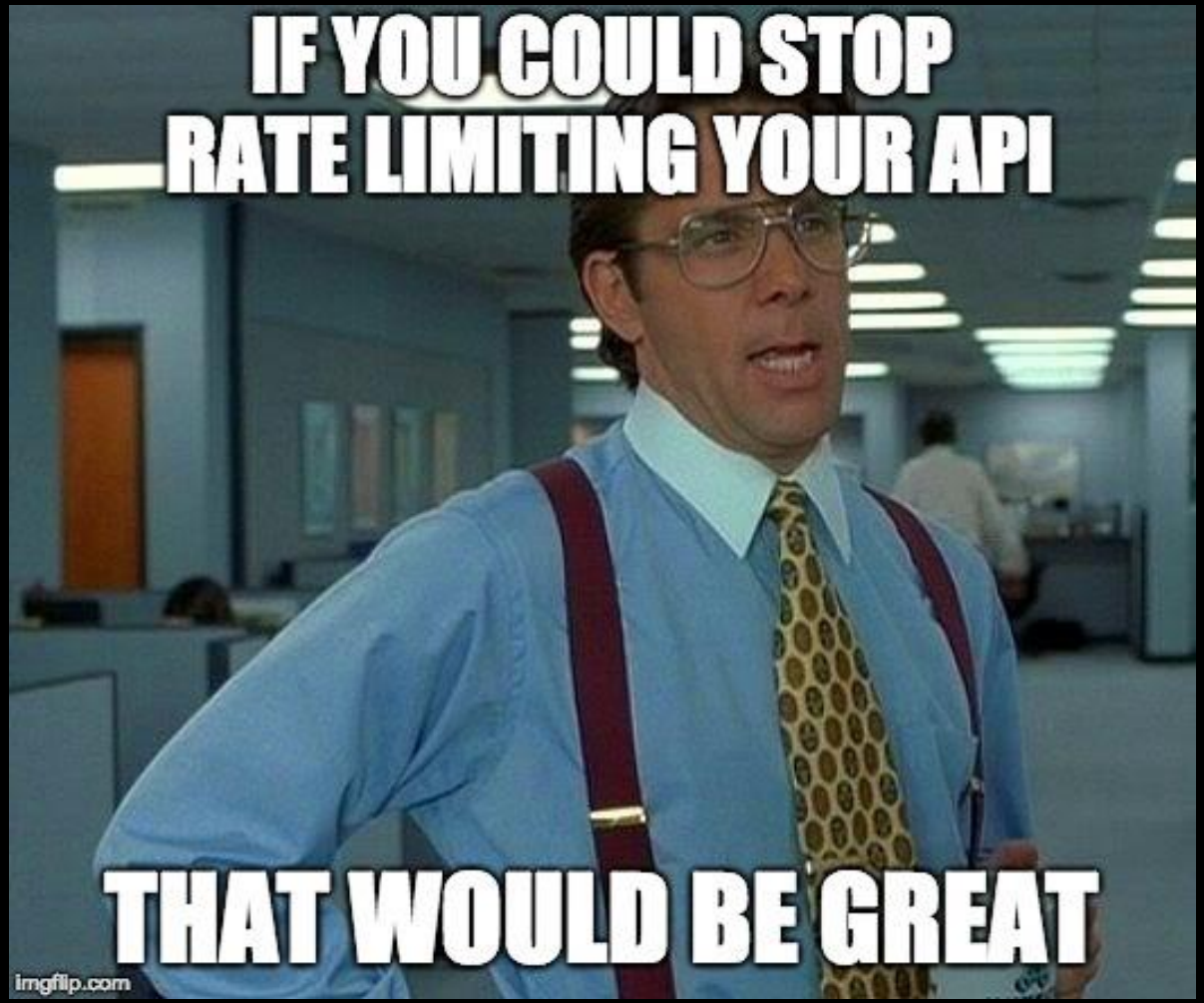
1. Checkout (ie. Download) code Repository
2. Branch, Apply Diff, & Commit
3. Fork Repository on GitHub
4. Rename Repository on GitHub
5. Push changes
6. Create Pull Request on GitHub

File IO Git Operation GitHub API

Pull Request Generation Steps

1. Checkout (ie. Download) code Repository
2. Branch, Apply Diff, & Commit
3. Fork Repository on GitHub
4. Rename Repository on GitHub
5. Push changes
6. Create Pull Request on GitHub

File IO | Git Operation | GitHub API



We’ve made it this far

- [x] Vulnerabilities Detected
- [x] Style Detected
- [x] Code Fixed & Diff Generated
- [x] Rate Limit Bypassed

We’ve made it this far

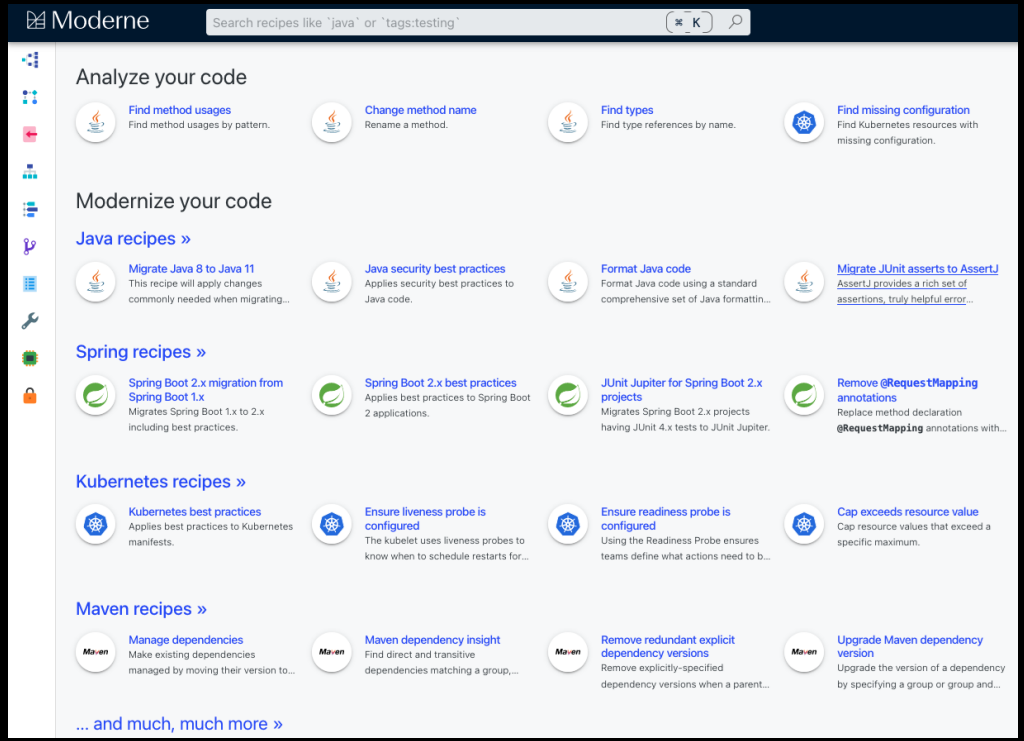
- [x] Vulnerabilities Detected
- [x] Style Detected
- [x] Code Fixed & Diff Generated
- [x] Rate Limit Bypassed

How do we do this for all the repositories?

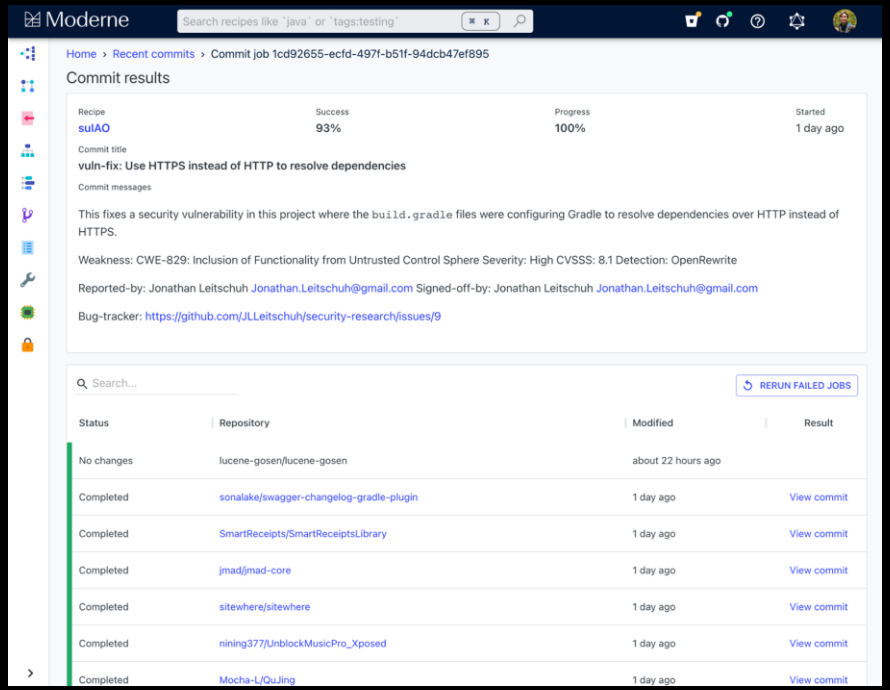
Moderne

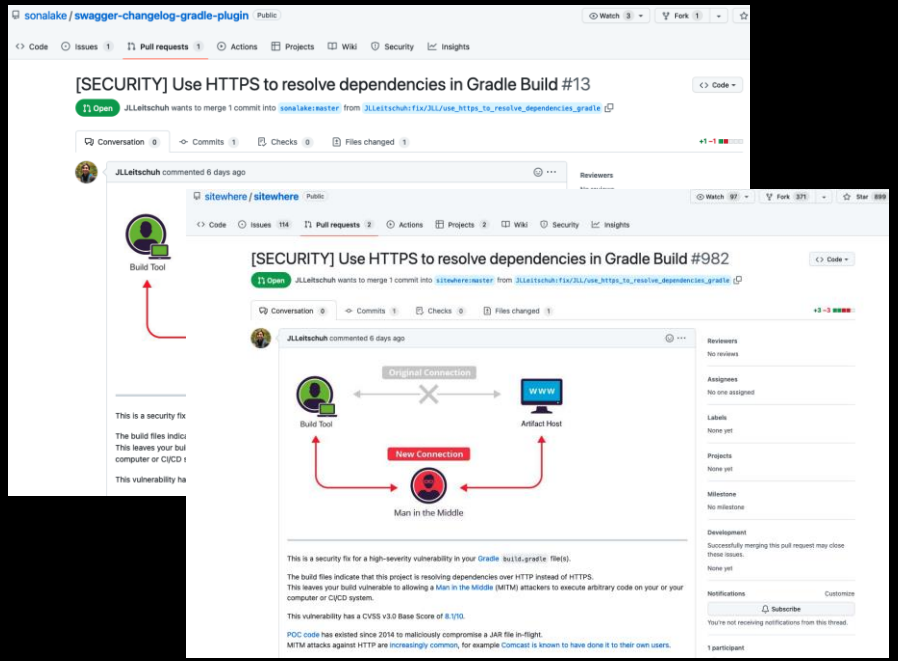
- Free for Open Source Projects!
- ~7,000 Repositories indexed
- Run Open Rewrite Transformations at Scale
- Generates and Updates Pull Requests

800+ OpenRewrite Recipes including complete Framework Migrations



Bulk Pull Request Generation - public.moderne.io





Use Files#createTempDirectory

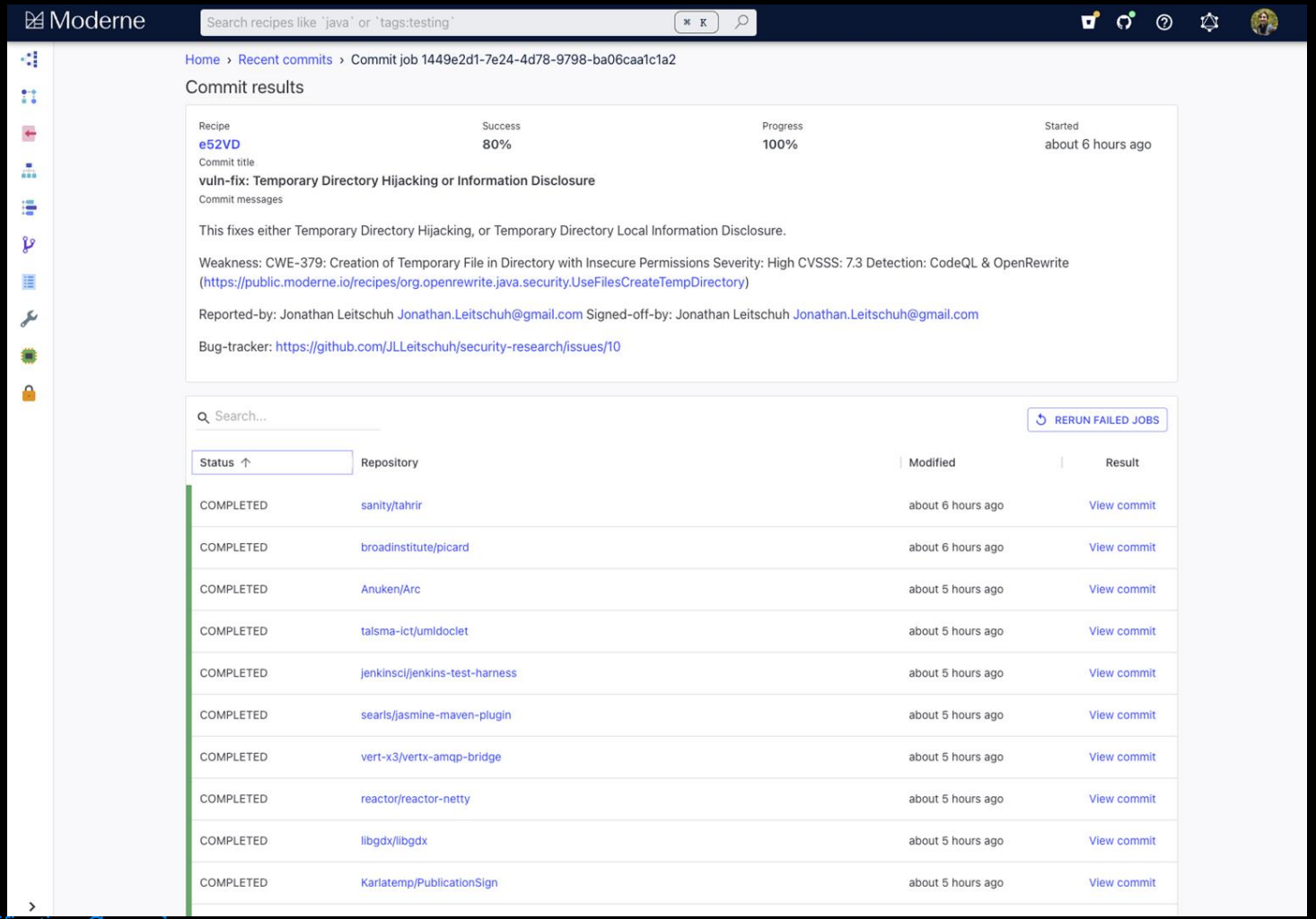
REPLAY

SUMMARY

Status	Started	Estimated time savings	Repositories searched	Repositories changed	Files searched	Files changed
Finished	about 2 hours ago	17 hours, 50 minutes	6.21k	80	1.26M	107

Search... Hide no results SELECT ALL WITH RESULTS

Status	Repository	Branch	Total results	Files searched	Actions
Finished	broadinstitute/picard	master	4	6,501	DIFF
Finished	jenkinsci/google-oauth-plugin	develop	3	218	DIFF
Finished	andyglick/jenesis4java	master	3	279	DIFF
Finished	salesforce/ImageOptimization	master	3	244	DIFF
Finished	sonatype/plexus-archiver	master	3	580	DIFF
Finished	zeroturnaround/zt-zip	master	3	230	DIFF
Finished	jenkinsci/acceptance-test-harness	master	3	786	DIFF
Finished	jenkinsci/jacoco-plugin	master	3	333	DIFF
Finished	smacke/jaydio	master	2	154	DIFF
Finished	jaltekruse/OpenNotebook	master	2	438	DIFF
Finished	nativelibs4java/BridJ	master	2	627	DIFF
Finished	jenkinsci/backend-jpi-create	master	2	134	DIFF
Finished	libgdx/libgdx	master	2	3,718	DIFF
Finished	koraktor/mavanagaiata	master	2	75	DIFF
Finished	Graylog2/JadConfig	master	2	357	DIFF
Finished	jbossas/jboss-vfs	master	2	279	DIFF



But there are more than just 7,000 repositories in the world

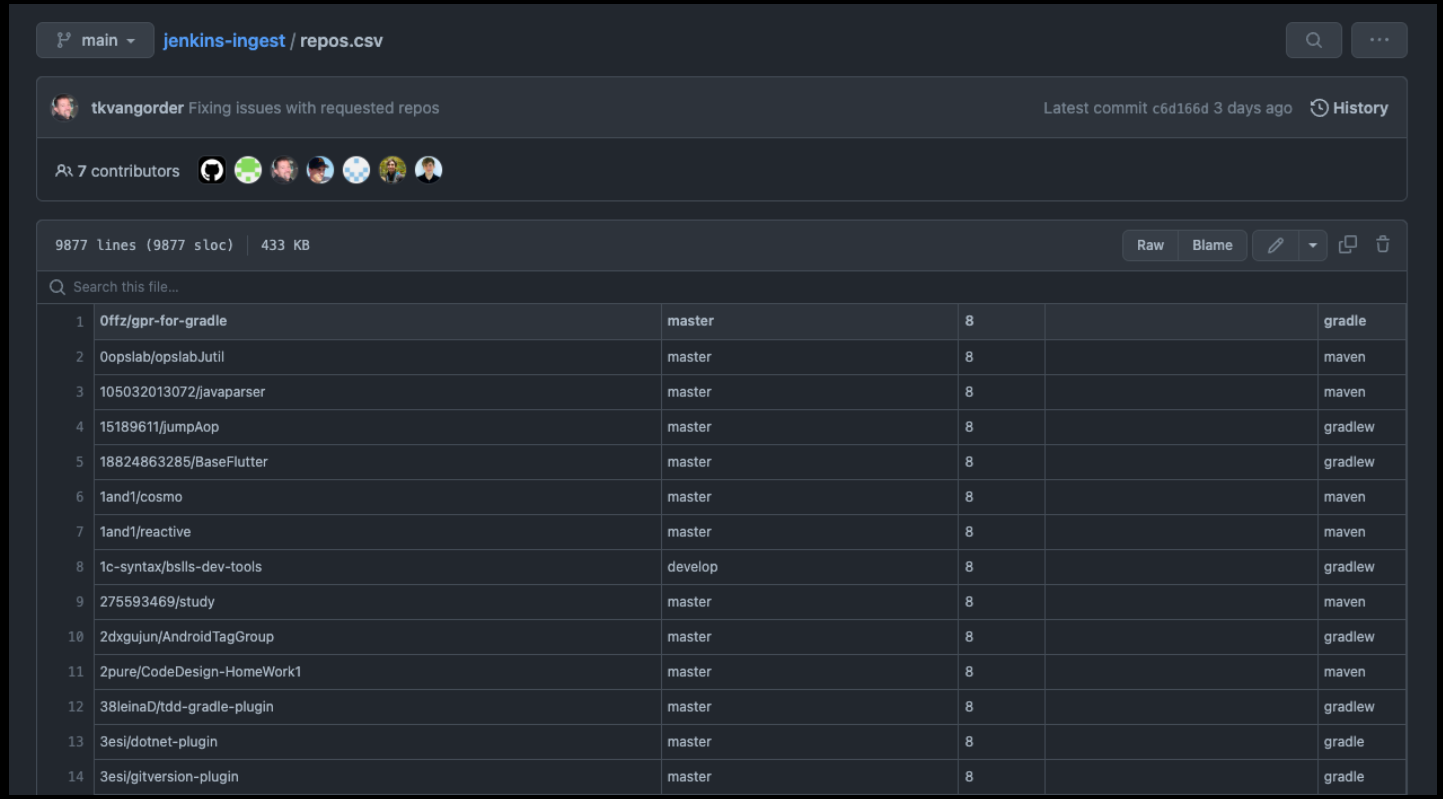
How do we find the other vulnerable projects?

CodeQL

CodeQL

100k+ OSS Projects Indexed
35k+ OSS Java Projects

https://github.com/moderneinc/jenkins-ingest



1	0ffz/gpr-for-gradle	master	8		gradle
2	0opslab/opslabJutil	master	8		maven
3	105032013072/javaparser	master	8		maven
4	15189611/jumpAop	master	8		gradlew
5	18824863285/BaseFlutter	master	8		gradlew
6	1and1/cosmo	master	8		maven
7	1and1/reactive	master	8		maven
8	1c-syntax/bsls-dev-tools	develop	8		gradlew
9	275593469/study	master	8		maven
10	2dxgujun/AndroidTagGroup	master	8		gradlew
11	2pure/CodeDesign-HomeWork1	master	8		maven
12	38leinaD/tdd-gradle-plugin	master	8		gradlew
13	3esi/dotnet-plugin	master	8		gradle
14	3esi/gitversion-plugin	master	8		gradle

Finally!

Let’s generate some
Open Source Software
Pull Requests!

Bulk Pull Request Generation Statistics

Project	PR Generator	Pull Requests	Merge Rate
HTTP Download of Dependencies	Python Bot	1,596	40%
CVE-2019-16303: JHipster RNG Vulnerability	Python Bot + Moderne	3,467	2.3%
CVE-2020-8597: rhostname array overflow	Python Bot	1,885	7.6%
Temporary Directory Hijacking	Moderne	64	TBD
Partial Path Traversal	Moderne	32	TBD
Zip Slip	Moderne	100	TBD

Bulk Pull Request Generation Statistics

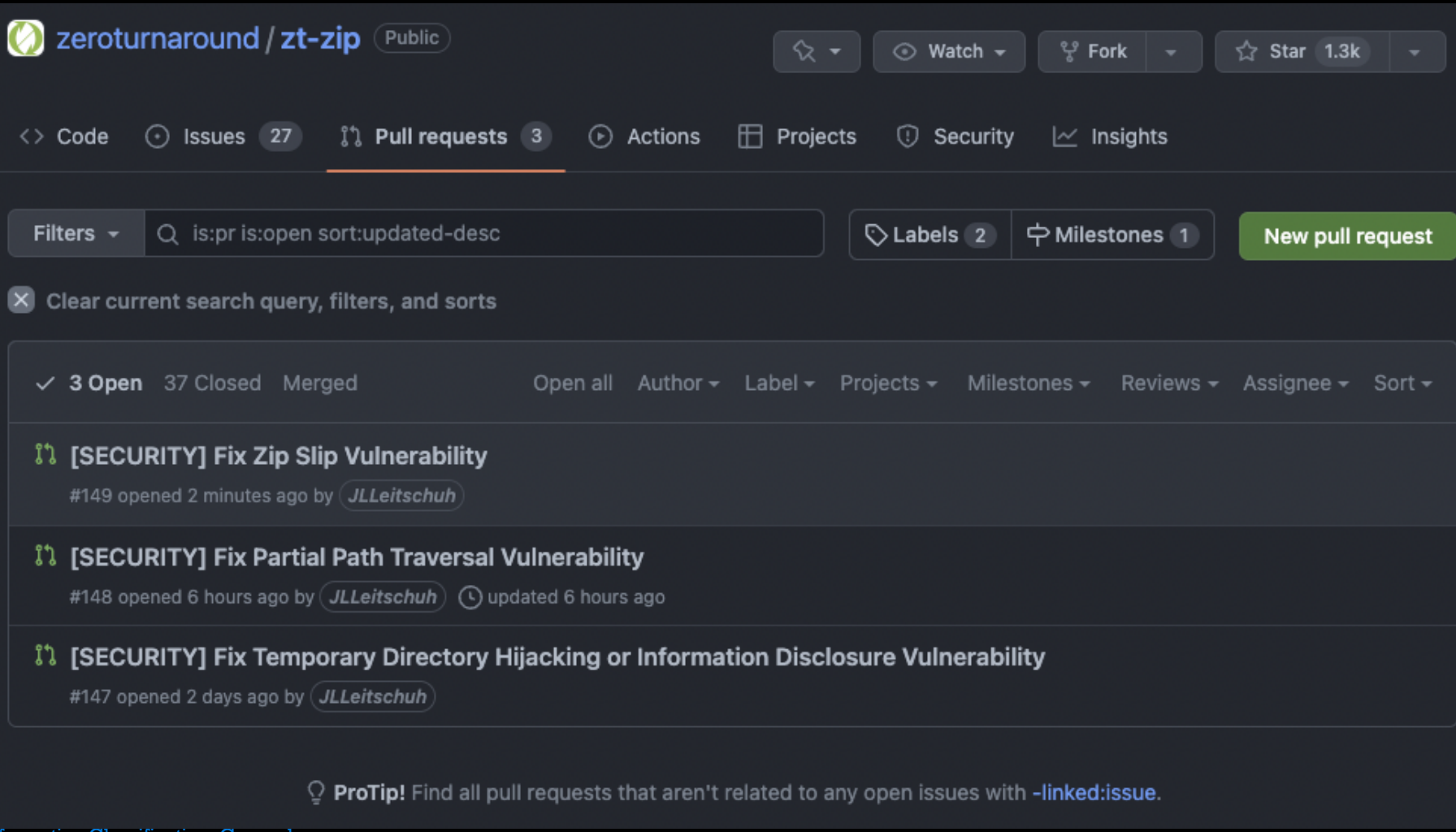
Project	PR Generator	Pull Requests	Merge Rate
HTTP Download of Dependencies	Python Bot	1,596	40%
CVE-2019-16303: JHipster RNG Vulnerability	Python Bot + Moderne	3,467	2.3%
CVE-2020-8597: rhostname array overflow	Python Bot	1,885	7.6%
Temporary Directory Hijacking	Moderne	64	TBD
Partial Path Traversal	Moderne	32	TBD
Zip Slip	Moderne	100	TBD

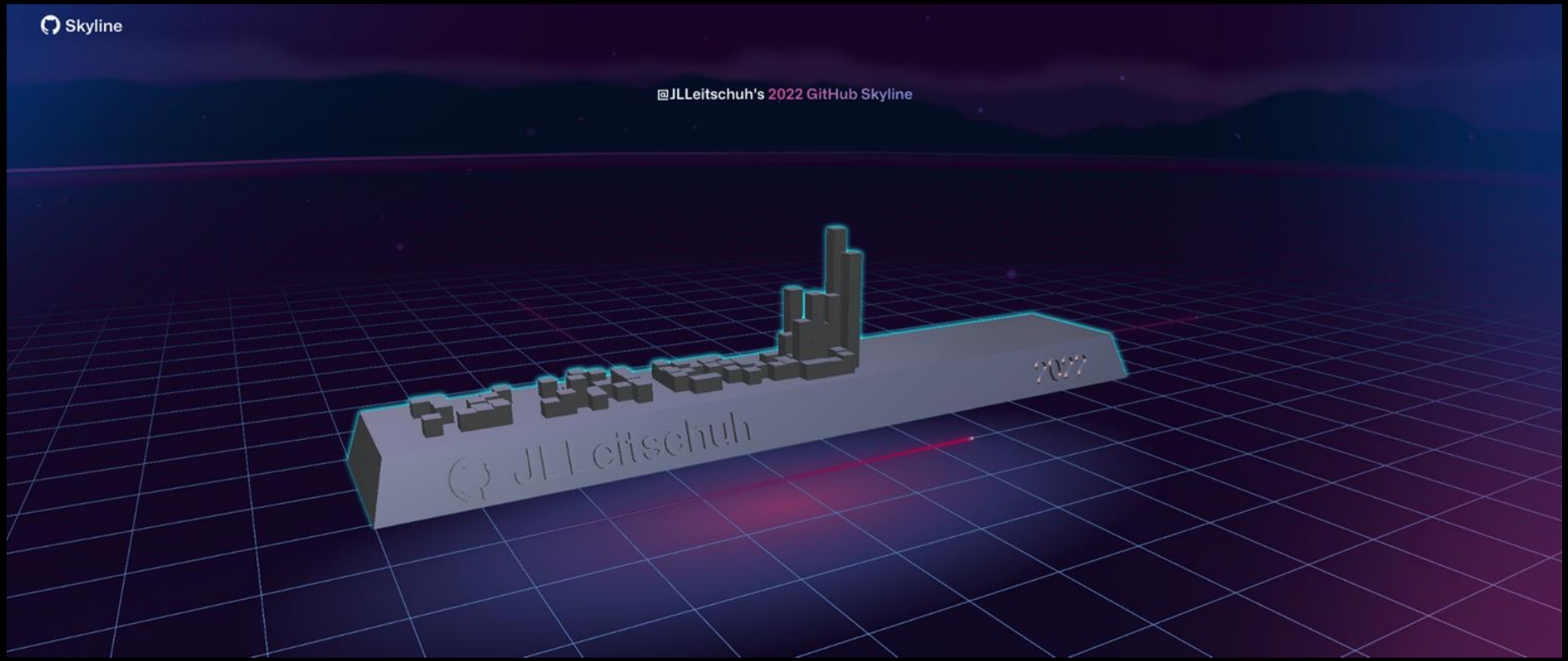
New Pull Requests Generated in 2022: 590+

Bulk Pull Request Generation Statistics

Project	PR Generator	Pull Requests	Merge Rate
HTTP Download of Dependencies	Python Bot	1,596	40%
CVE-2019-16303: JHipster RNG Vulnerability	Python Bot + Moderne	3,467	2.3%
CVE-2020-8597: rhostname array overflow	Python Bot	1,885	7.6%
Temporary Directory Hijacking	Moderne	64	TBD
Partial Path Traversal	Moderne	32	TBD
Zip Slip	Moderne	100	TBD

Personally Generated: 5,200+ Pull Requests





Best Practices for Bulk Pull Request Generation

Messaging!

All Software Problems are
People Problems
In Disguise

Lesson 1

Sign off all Commits

--signoff

Sign off on Commits

Signed-off-by: Jonathan Leitschuh <Jonathan.Leitschuh@gmail.com>

Sign off on Commits

Why?!

Sign off on Commits

“It was introduced in the wake of the SCO lawsuit, (and other accusations of copyright infringement from SCO, most of which they never actually took to court), as a Developers Certificate of Origin. It is used to say that you certify that you have created the patch in question, or that you certify that to the best of your knowledge, it was created under an appropriate open-source license, or that it has been provided to you by someone else under those terms.”

- Stack Overflow

TL;DR

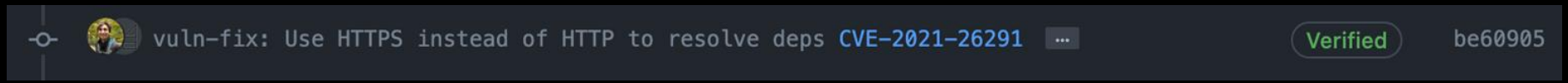
Lawyers

Lesson 2

Be a good commitizen

Lesson 2

Be a good commitizen
GPG Sign your Commits



Enjoy!

Browse files

master

torvalds committed on Aug 4, 2015 0 parents commit 9b0562595cc479ac8696110cb0a2d33f8f2b7d29 patch diff

No Whitespace Showing **1 changed file** with **10 additions** and **0 deletions.** Split Unified

10 README.md

```
@@ -0,0 +1,10 @@
Instructions on masquerading as other users in git:

```bash
export GIT_AUTHOR_NAME="Linus Torvalds"
export GIT_AUTHOR_EMAIL="torvalds@linux-foundation.org"
export GIT_COMMITTER_NAME="$GIT_AUTHOR_NAME"
export GIT_COMMITTER_EMAIL="$GIT_AUTHOR_EMAIL"

git commit -m "Enjoy!"
```
```

Lesson 3

SECOM
Commit Format

SECOM

```
vuln-fix: subject/header containing summary of changes in ~50 characters (Vuln-ID,)

Detailed explanation of the subject/header in ~75 words.
(what) Explain the security issue(s) that this commit is patching.
(why) Focus on why this patch is important and its impact.
(how) Describe how the issue is patched.

[For Each Weakness in Weaknesses:]
Weakness: weakness identification or CWE-ID.
Severity: severity of the issue (Low, Medium, High, Critical).
CVSS: numerical representation (0-10) of the vulnerability severity.
Detection: method used to detect the issue (Tool, Manual, Exploit).
Report: http://link-to-report/
Introduced in: commit hash.
[End]

Reported-by: reporter name 1 <reporter-email-1@host.com>
Reported-by: reporter name 2 <reporter-email-2@host.com>
Signed-off-by: your name <your-email@yourhost.com>

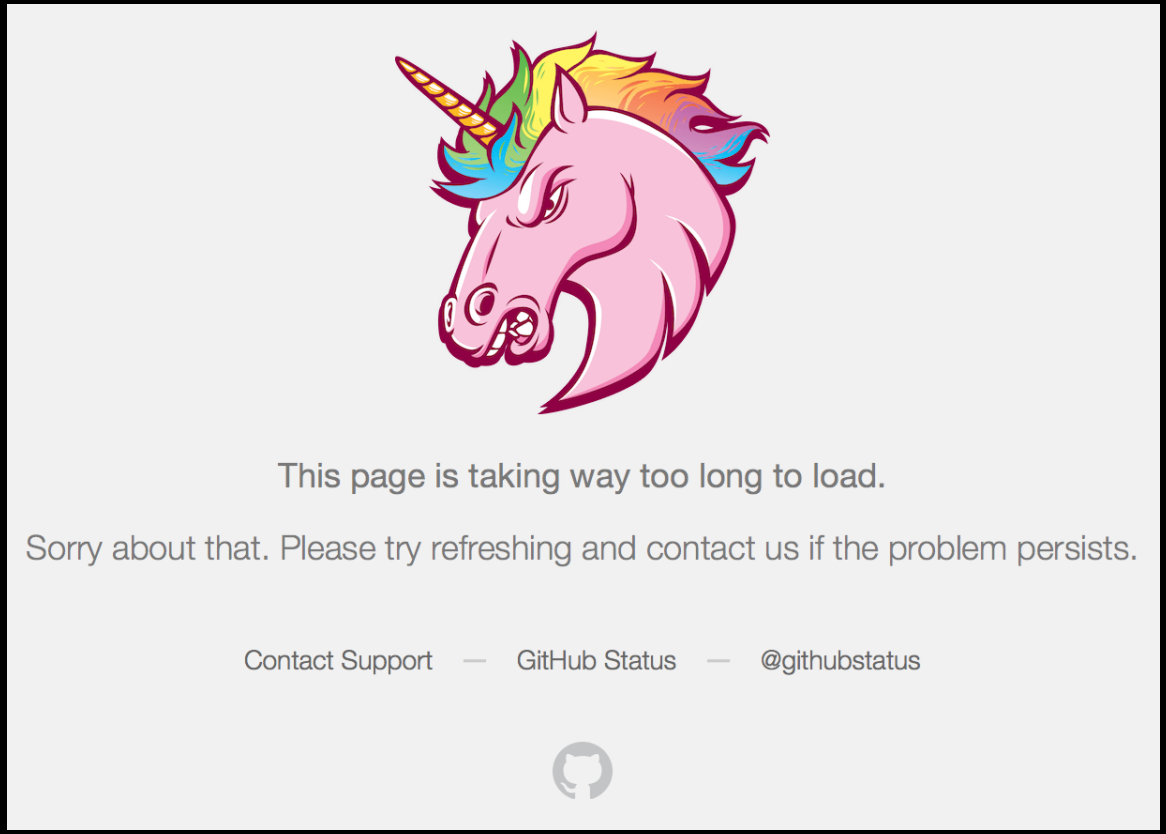
[If you use an issue tracker, add reference to it here:]
[if external issue tracker:]
Bug-tracker: https://link-to-bug-tracker/id

[if github used as issue tracker:]
Resolves: #123
See also: #456, #789
```

Lesson 4

There are risks using your personal GitHub Account

Anyone here familiar with
GitHub’s
Angry Unicorn?



This was my GitHub Profile Page for most of 2020

Lesson 5

Coordinate with GitHub

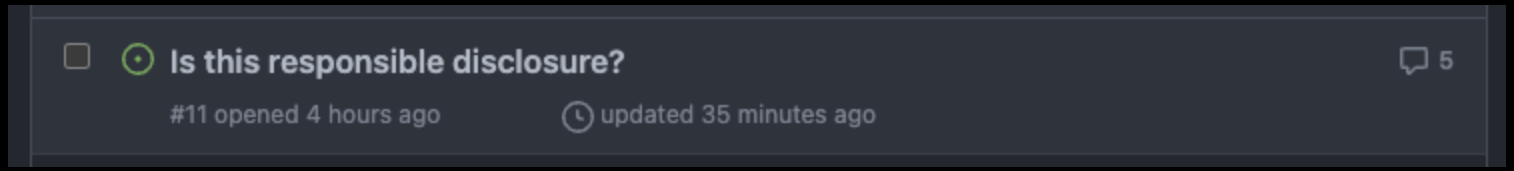
Before Attempting

Reach out to GitHub!

SecurityLab@github.com

Lesson 5

Consider the Implications



Conclusion

As Security Researchers

We have an obligation to society

We know these vulnerabilities are out there

“For every 500 developers you have one security researcher.”

- GitHub 2020

> “We can fix it. We have the technology. OK. We need to create the technology. Alright. The policy guys are mucking with the technology. Relax. WE'RE ON IT.

- Dan Kaminsky (1979 – 2021)

Sound Bytes

- Learn CodeQL! Seriously! It’s an incredibly powerful language!
- Contribute to OpenRewrite! Deploy your security fixes at scale!
- Join the GitHub Security Lab & OpenRewrite Slack Channels!

Thanks

Lidia Giuliano

Shyam Mehta