black hat®
EUROPE 2020
DECEMBER 9–10
BRIEFINGS

Effective Vulnerability Discovery with Machine Learning

Asankhaya Sharma
Veracode

Ang Ming Yi
Veracode

We’re going to talk about vulnerability discovery

- Finding vulnerabilities in your application goes beyond code you have written
- It even goes beyond the libraries directly required by your code
- More libraries means a wider surface of attack
- We will discuss a way to discover these vulnerabilities at scale

About us

Asankhaya Sharma

Director of Engineering
Veracode

Ang Ming Yi

Senior Research Engineer
Veracode

Agenda

- Vulnerability Curation
- Machine Learning approach to Identify Vulnerabilities
- Effective Vulnerability Discovery

Difficulty in tracking down vulnerabilities

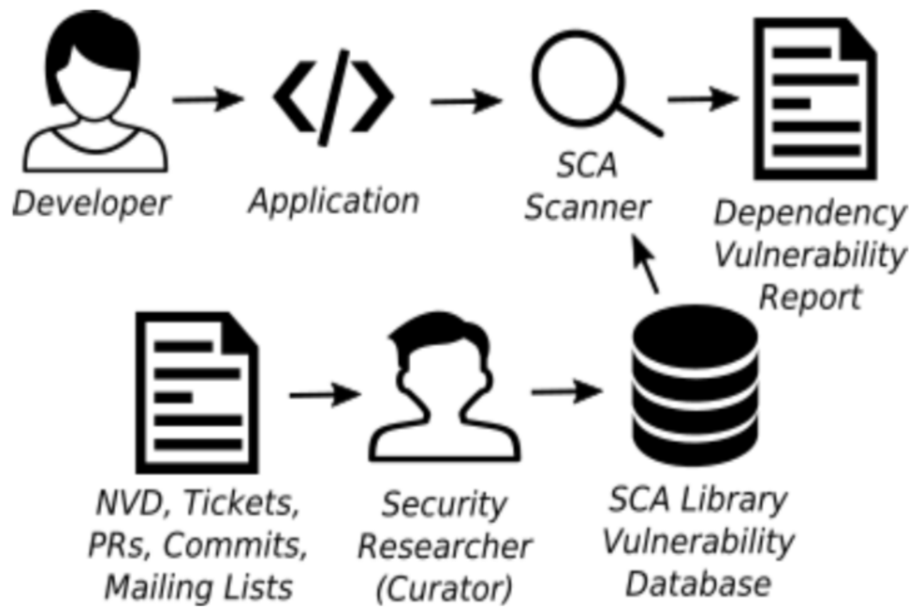
Software Applications often depends on and are built with Open-source libraries

- Software vulnerabilities exists in these third-party libraries
- Need to be aware of issues on both first-party and third-party code
- Difficult to track every component and vulnerability
- Need to curate vulnerabilities found in open-source libraries

How we can curate vulnerabilities

- Process data from various *internet sources:*

- National Vulnerability Database (NVD),
  JIRA tickets, Bugzilla issues,
  GitHub issues, GitHub pull requests,
  Git commits, Mailing lists,
  Vendor Release Notes


Developer
Application
SCA Scanner
Dependency Vulnerability Report
NVD, Tickets, PRs, Commits, Mailing Lists
Security Researcher (Curator)
SCA Library Vulnerability Database

Curating Vulnerabilities is difficult

- Curation process is manual
- Support for over 2.6 million open-source libraries
- Actively track >50,000 Git Repositories, and more…
- Inelastic resources – Security Researchers
- We needed a solution to scale the system

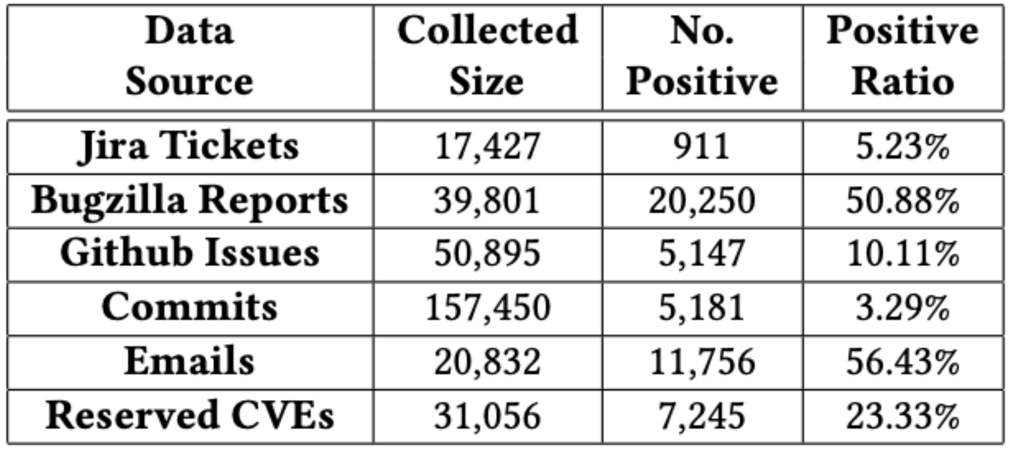
A naturally highly imbalanced dataset

- Highly Imbalanced Ratio per source:
  - As low as 5.88% labeled vulnerability,
  - As high as 41.42% labeled not a vulnerability
- Continue to expand on sources
  - Original data based on ~20k repositories << New data ~50k repositories
  - Extended language and library coverage
- Labeled data is now a subset of the set of positively predicted data
- We needed a solution to balance, and scale, the system

The Machine Learning Approach

Goal: To automatically generate improved, and evaluated, models resilient to changes in requirements

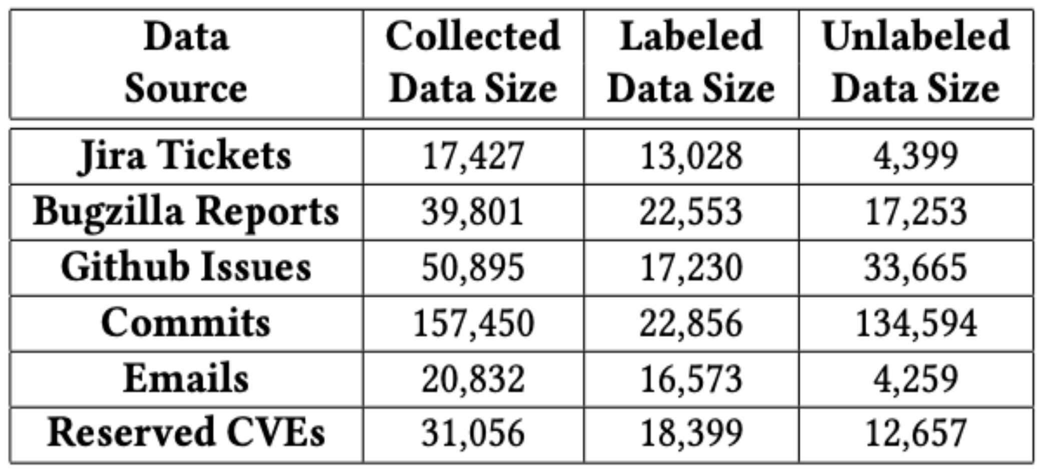
- Incorporate more data sources, more language support
- Dataset has become highly imbalanced (Before: 5.88%, Current: 3.29%)
- Unused unlabelled data has piled up
- New approach presented at Mining Software Repositories (MSR) 2020



Data Source	Collected Size	No. Positive	Positive Ratio
Jira Tickets	17,427	911	5.23%
Bugzilla Reports	39,801	20,250	50.88%
Github Issues	50,895	5,147	10.11%
Commits	157,450	5,181	3.29%
Emails	20,832	11,756	56.43%
Reserved CVEs	31,056	7,245	23.33%

Solving data imbalance issue with Self-Training

- Issues
  - Unlabelled data >>> Labelled data
  - Imbalanced data
- Researchers only see data predicted as vulnerable, some of the data predicted as *non-vulnerable* can be informative
- Further, a portion of data which have never passed through the initial filter before the machine learning service, remains unlabelled.
- Utilise this unlabelled data using self-training [Nigam et al.]

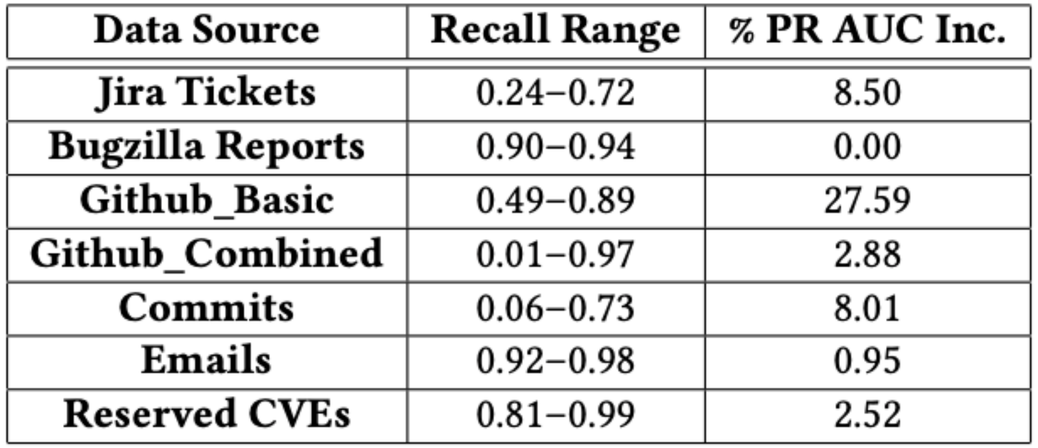


Data Source	Collected Data Size	Labeled Data Size	Unlabeled Data Size
Jira Tickets	17,427	13,028	4,399
Bugzilla Reports	39,801	22,553	17,253
Github Issues	50,895	17,230	33,665
Commits	157,450	22,856	134,594
Emails	20,832	16,573	4,259
Reserved CVEs	31,056	18,399	12,657

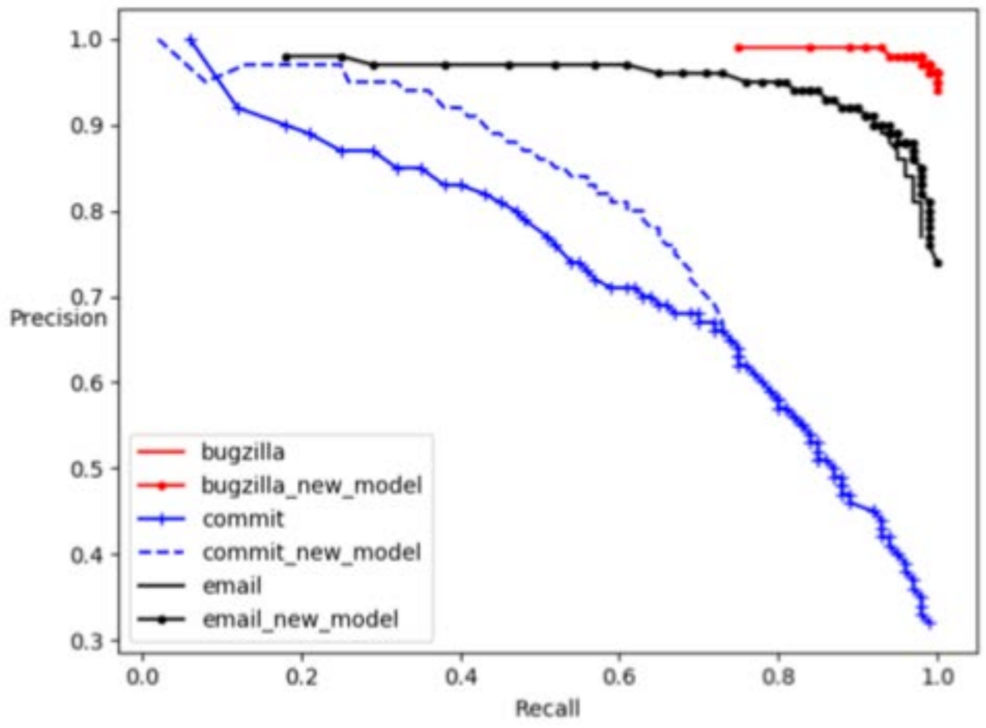
Evaluating the Machine Learning Approach

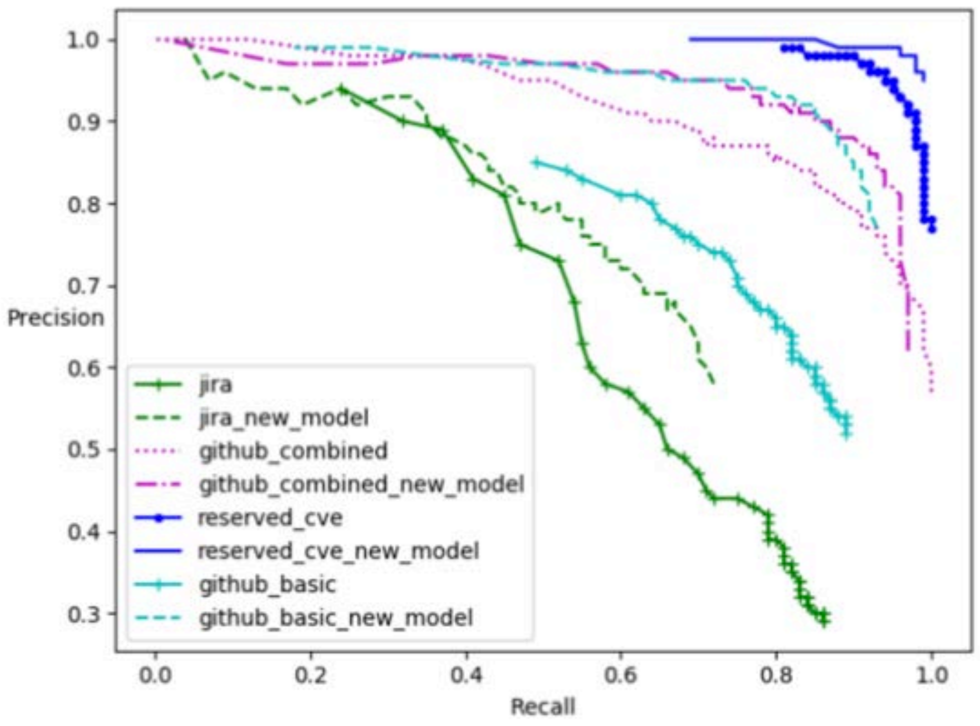
- Generally observed an increase in the Area Under Curve (AUC) graph

  (A higher AUC curve indicates higher 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





Efficient Vulnerability Discovery

- Machine Learning Approach is efficient, and only the first piece to the process
- What are we looking for?
- Why?
- Why don’t we just lookup Central Authorities/Vulnerability Databases?

Motivation

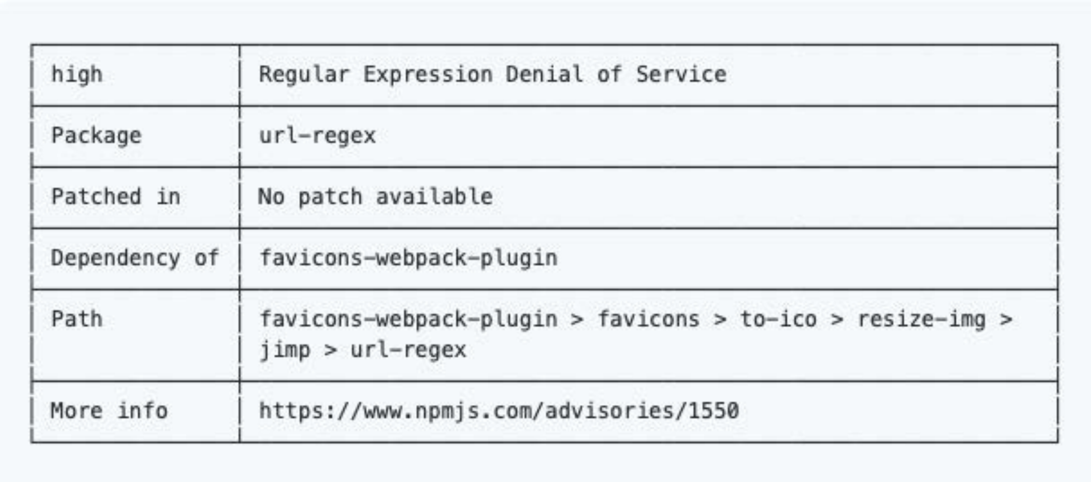
- The Nature of Modern Software Composition
- The Devil’s in the Dependencies
- Inefficiency of Central Authorities

How to ensure our code is secure?

- Safe coding practices – type checking, data validation, input validation, etc
- Static Analysis – Locating unwanted behavior, fixing insecure code
- Dynamic Analysis – Black-box testing, fuzzing
- Penetration tests – Find other security flaws

Potential Flaws gained by ignoring our dependencies

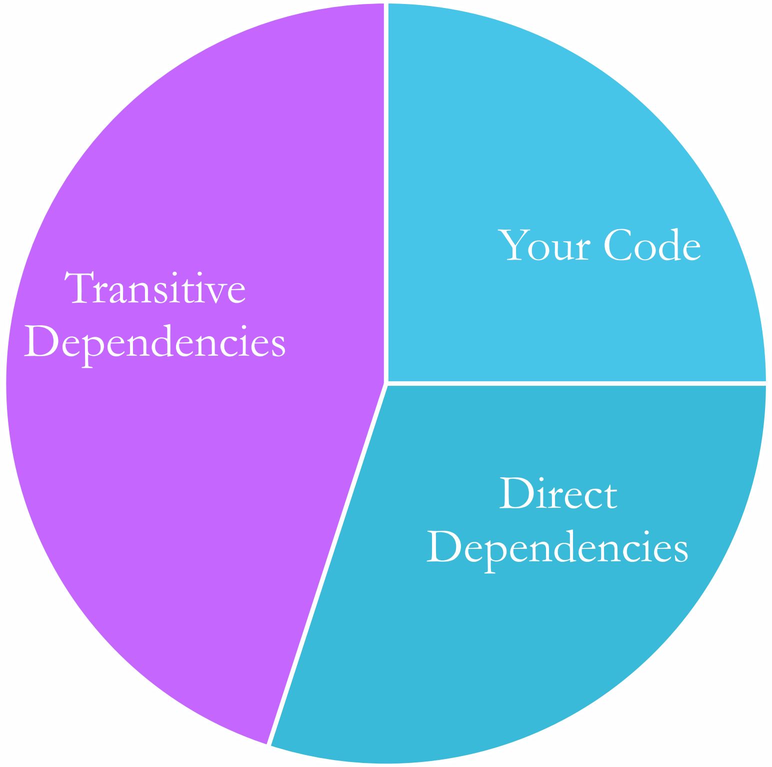
- Dependencies have code flaws too!
    - Cross-site Scripting, Arbitrary Code Execution, Deserialization flaws, Directory Traversal, Denial of Service, Man-in-the-Middle, etc
- Dependencies can also introduce external threats
    - Malicious Code Injection
    - Malicious pre-install/post-install exfiltration scripts
    - Malicious takeover of legit packages (eg. eslint-scope, purescript-installer)
    - Numerous amounts of typosquatters (eg. atlas_client vs atlas-client; jellyfish vs jeIlyfish)



high	Regular Expression Denial of Service
Package	url-regex
Patched in	No patch available
Dependency of	favicons-webpack-plugin
Path	favicons-webpack-plugin > favicons > to-ico > resize-img > jimp > url-regex
More info	https://www.npmjs.com/advisories/1550

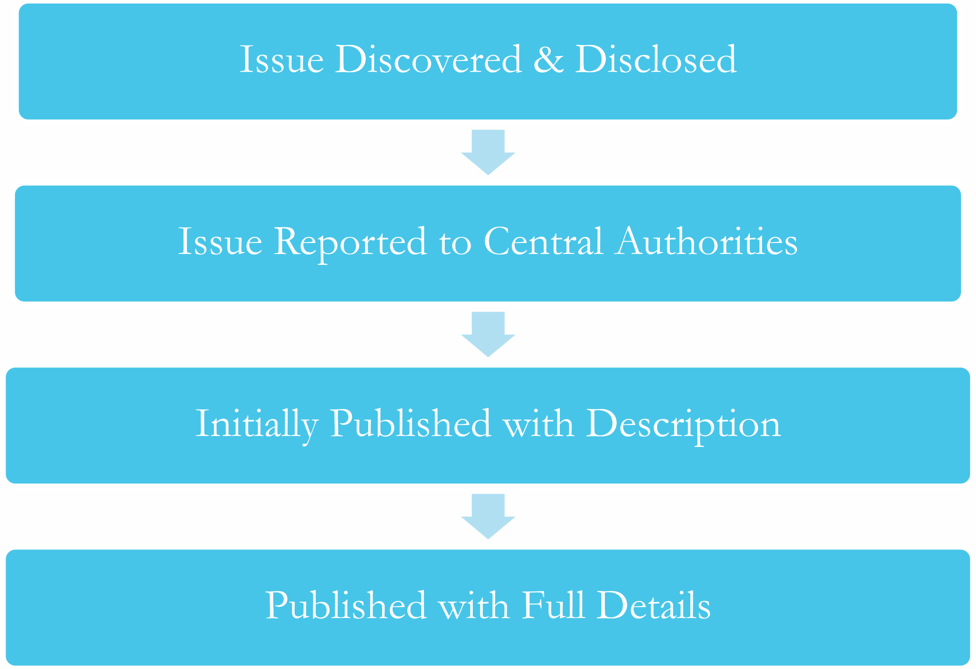
The Nature of Modern Software Composition

- Software built with third party libraries (eg. Spring Boot, requests, jquery, js-yaml)
- Number of third party libraries used in Real-world application varies
  - Typically ranges from 10s to 100s, even 1000s of third party libraries used
  - Top 10 used libraries in Javascript included in >80% of Javascript Applications
  - ~70% of the applications tested has at least 1 external library flaw, and
  - >46% of these libraries are only pulled in transitively



Inefficiency of Central Authorities

- Time taken for vulnerabilities to be published from initial disclosure
- Incompleteness and/or Imprecision of data
    - “Affects all version before Version X”
    - “Spring Boot” vs. “spring-boot-loader-tools” (CVE-2018-1196)
- There are flaws not found on Central Authorities
    - Varies per language
    - Overall ~15%



Issue Discovered & Disclosed

Issue Reported to Central Authorities

Initially Published with Description

Published with Full Details

Intuition of discovering vulnerabilities

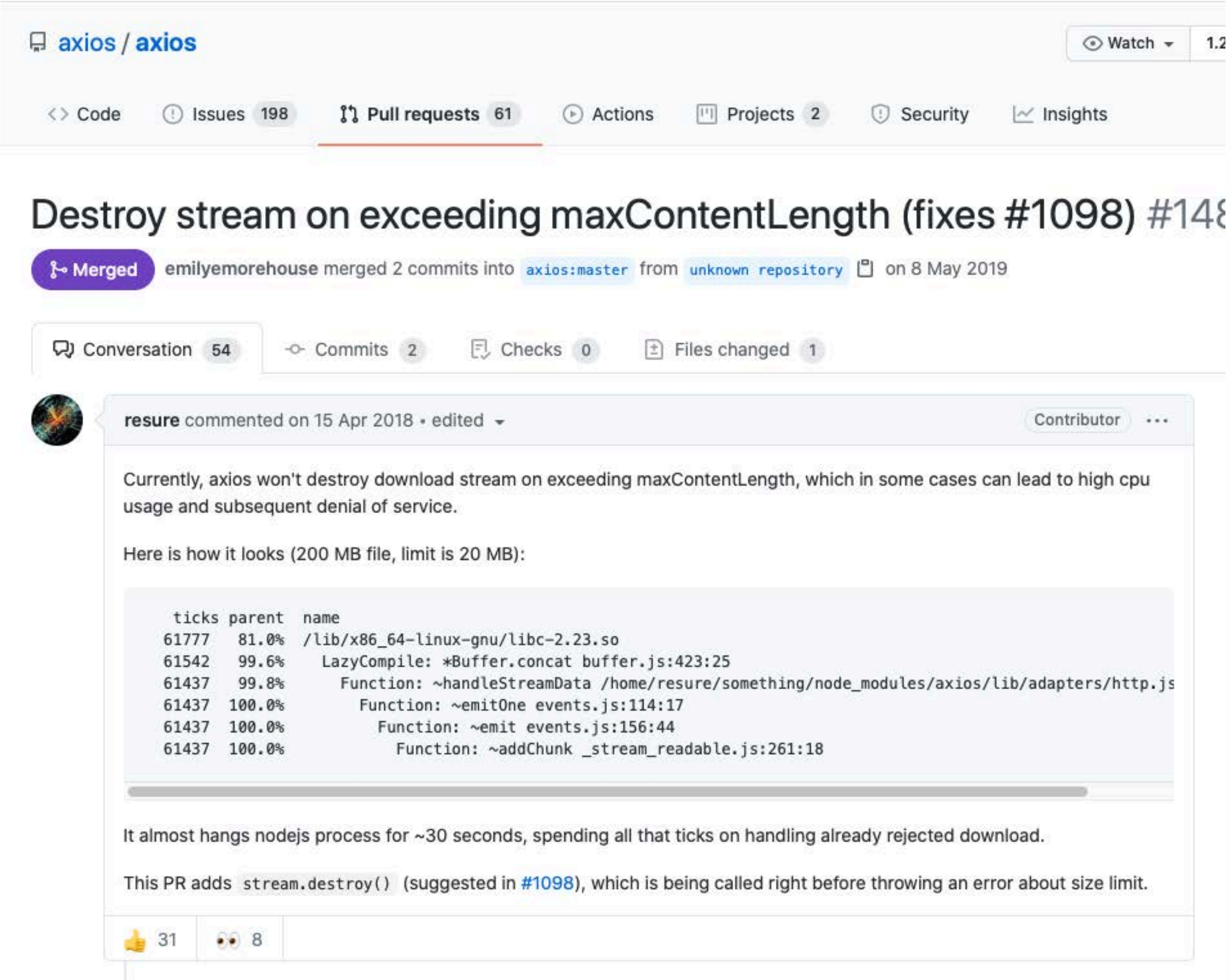
- Resources are limited – Inefficient and expensive to vet through every single line of code, depending on complexity,
  - Teams may barely keep up with static scans of first party code
  - Fuzzing may take weeks or months
  - Penetration testing cannot be done frequent enough
- New libraries/dependencies releases can invalidate previous findings; Updates are usually many and frequent

Where should we start looking from?

- Not static/dynamic scans
- As close as possible to where developers, contributors, would interact, eg:
  - GitHub
  - JIRA
  - Bugzilla
  - Mailing Lists
  - Release notes
- Can result in a large dataset, >100,000s weekly data; Back to the same challenges with limited resources
  - Machine Learning Approach reduces this amount to 1000s weekly

Examples of data we found useful

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



Destroy stream on exceeding maxContentLength (fixes #1098) #148

emilyemorehouse merged 2 commits into axios:master from unknown repository on 8 May 2019

resure commented on 15 Apr 2018 • edited

Currently, axios won't destroy download stream on exceeding maxContentLength, which in some cases can lead to high cpu usage and subsequent denial of service.

Here is how it looks (200 MB file, limit is 20 MB):

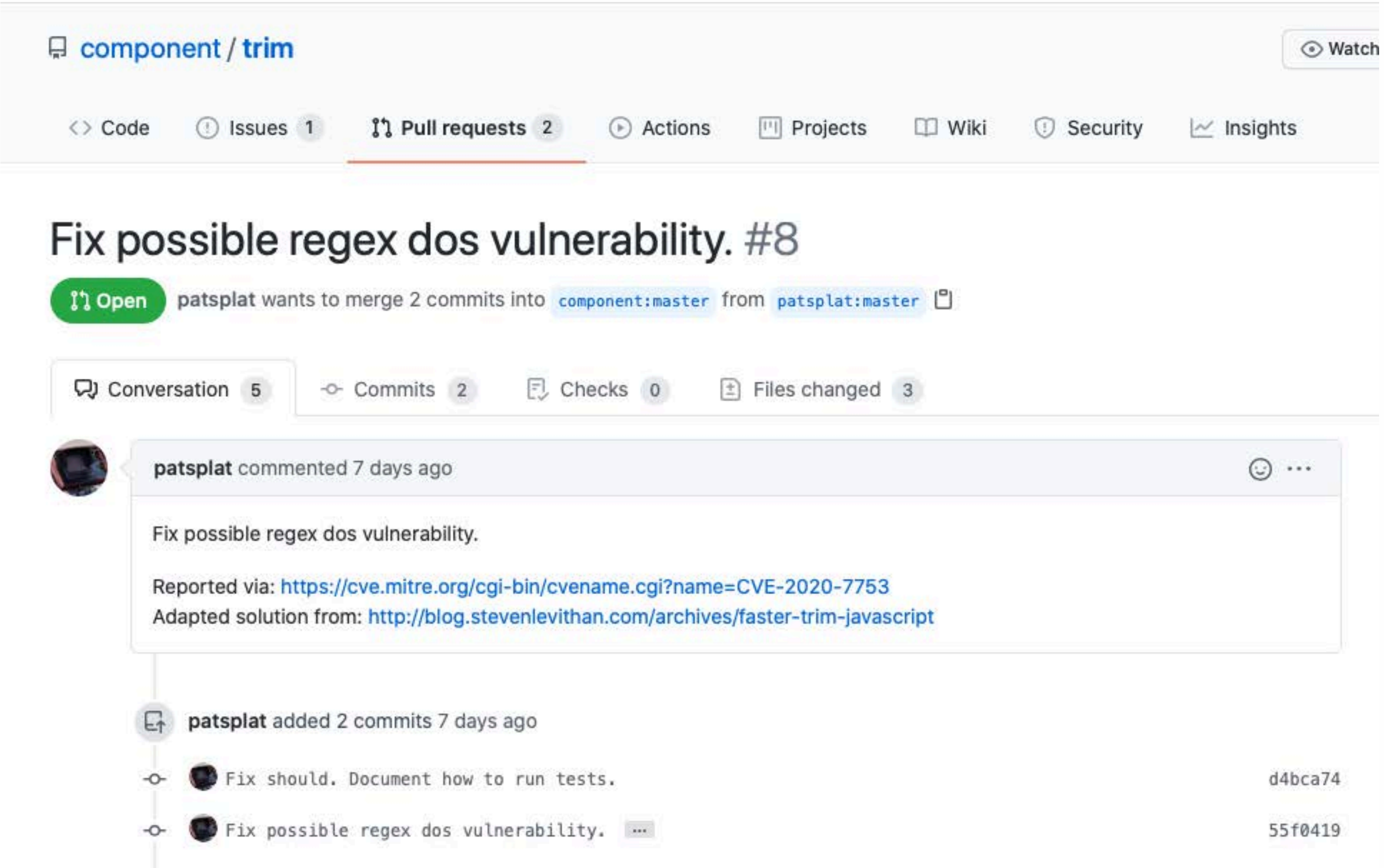
```
 ticks parent  name
61777   81.0%  /lib/x86_64-linux-gnu/libc-2.23.so
61542   99.6%    LazyCompile: *Buffer.concat buffer.js:423:25
61437   99.8%      Function: ~handleStreamData /home/resure/something/node_modules/axios/lib/adapters/http.js
61437  100.0%        Function: ~emitOne events.js:114:17
61437  100.0%          Function: ~emit events.js:156:44
61437  100.0%            Function: ~addChunk _stream_readable.js:261:18
```

It almost hangs nodejs process for ~30 seconds, spending all that ticks on handling already rejected download.

This PR adds `stream.destroy()` (suggested in #1098), which is being called right before throwing an error about size limit.

Examples of data we found useful

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



Fix possible regex dos vulnerability. #8

patsplat wants to merge 2 commits into `component:master` from `patsplat:master`

patsplat commented 7 days ago

Fix possible regex dos vulnerability.

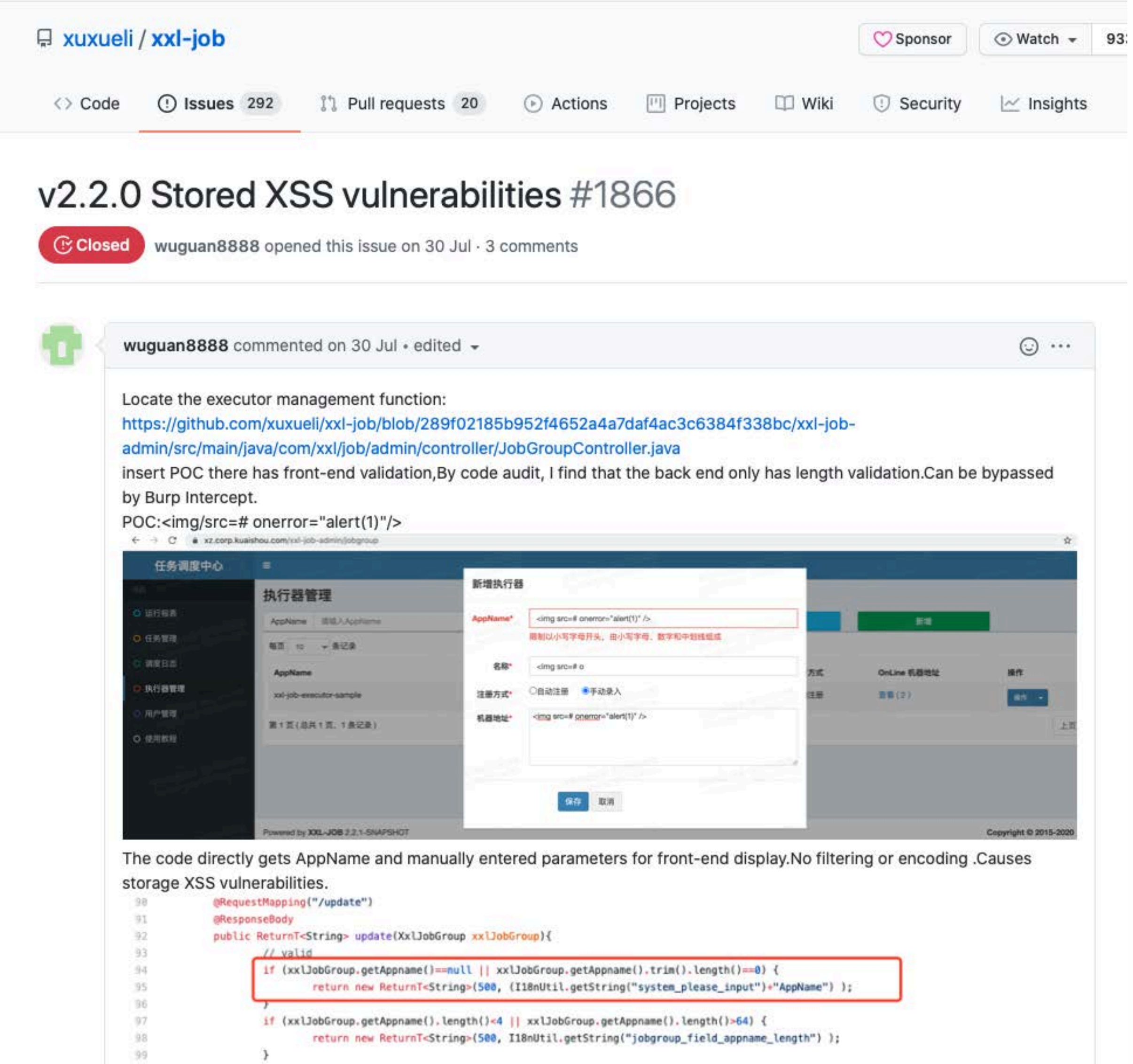
Reported via: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-7753
Adapted solution from: http://blog.stevenlevithan.com/archives/faster-trim-javascript

patsplat added 2 commits 7 days ago

```
Fix should. Document how to run tests.        d4bca74
Fix possible regex dos vulnerability. …       55f0419
```

Examples of data we found useful

- Persistent Cross-site Scripting (XSS)
- XSS is consistently on OWASP Top 10
- xxl-job
- >16k Stars
- Used by >2000 repositories
- 50 Contributors



xuxueli / xxl-job

v2.2.0 Stored XSS vulnerabilities #1866

Closed wuguan8888 opened this issue on 30 Jul · 3 comments

wuguan8888 commented on 30 Jul • edited

Locate the executor management function:
https://github.com/xuxueli/xxl-job/blob/289f02185b952f4652a4a7daf4ac3c6384f338bc/xxl-job-admin/src/main/java/com/xxl/job/admin/controller/JobGroupController.java
insert POC there has front-end validation,By code audit, I find that the back end only has length validation.Can be bypassed by Burp Intercept.
POC:<img/src=# onerror="alert(1)"/>

新增执行器

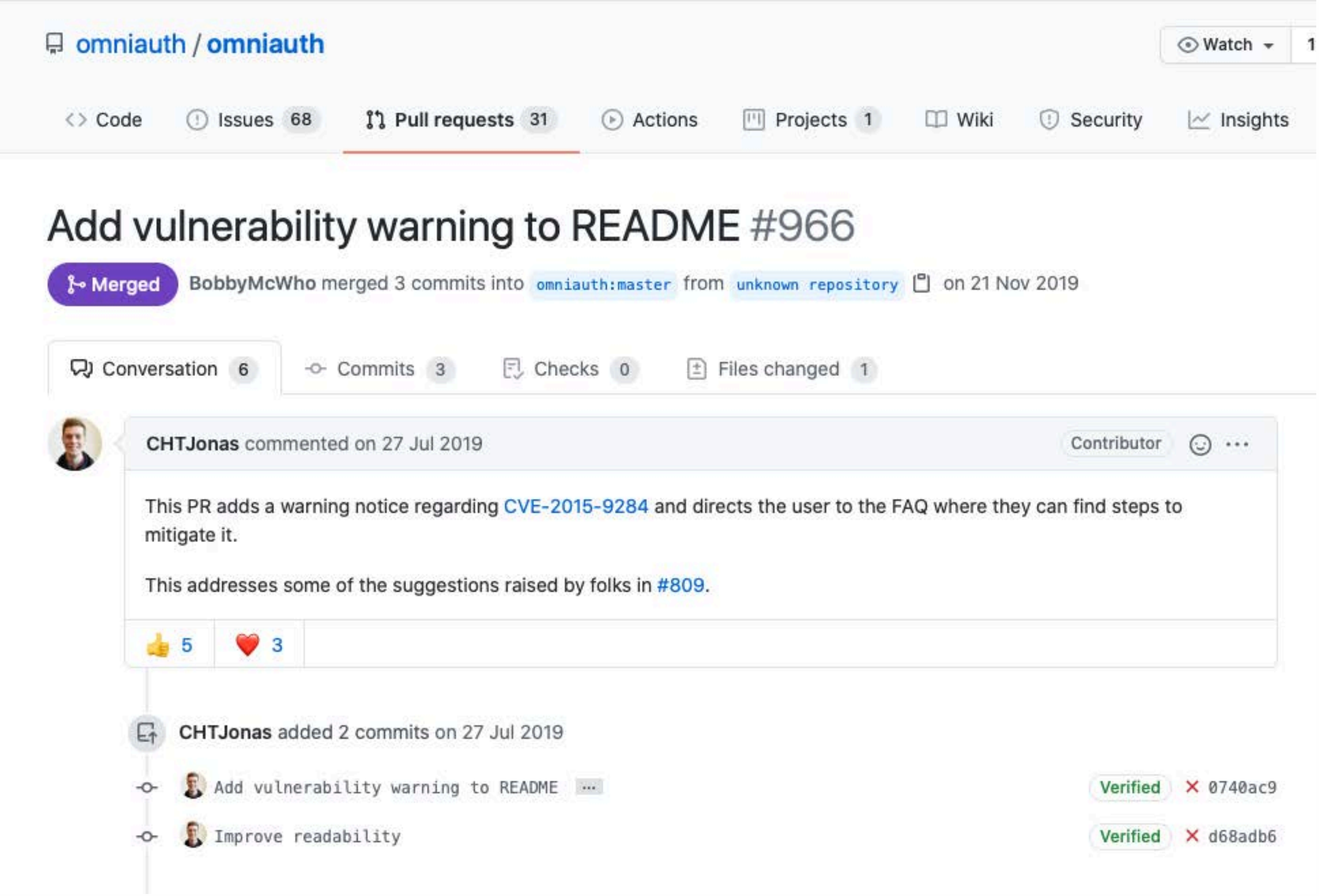
AppName*: <img src=# onerror="alert(1)" />
限制以小写字母开头，由小写字母、数字和中划线组成
名称*: <img src=# o
注册方式*: ○自动注册 ●手动录入
机器地址*: <img src=# onerror="alert(1)" />

The code directly gets AppName and manually entered parameters for front-end display.No filtering or encoding .Causes storage XSS vulnerabilities.

```
        @RequestMapping("/update")
        @ResponseBody
        public ReturnT<String> update(XxlJobGroup xxlJobGroup){
                // valid
                if (xxlJobGroup.getAppname()==null || xxlJobGroup.getAppname().trim().length()==0) {
                        return new ReturnT<String>(500, (I18nUtil.getString("system_please_input")+"AppName") );
                }
                if (xxlJobGroup.getAppname().length()<4 || xxlJobGroup.getAppname().length()>64) {
                        return new ReturnT<String>(500, I18nUtil.getString("jobgroup_field_appname_length") );
                }
```

Examples of data we found useful

- Cross-site Request Forgery (CSRF)
- CVE-2015-9284 (High Severity)
- omniauth
- Issue is not directly addressable by itself
- Applications have to mitigate it manually
- >55m Total downloads
- 170 Contributors



Add vulnerability warning to README #966

Merged BobbyMcWho merged 3 commits into `omniauth:master` from `unknown repository` on 21 Nov 2019

CHTJonas commented on 27 Jul 2019 — Contributor

This PR adds a warning notice regarding CVE-2015-9284 and directs the user to the FAQ where they can find steps to mitigate it.

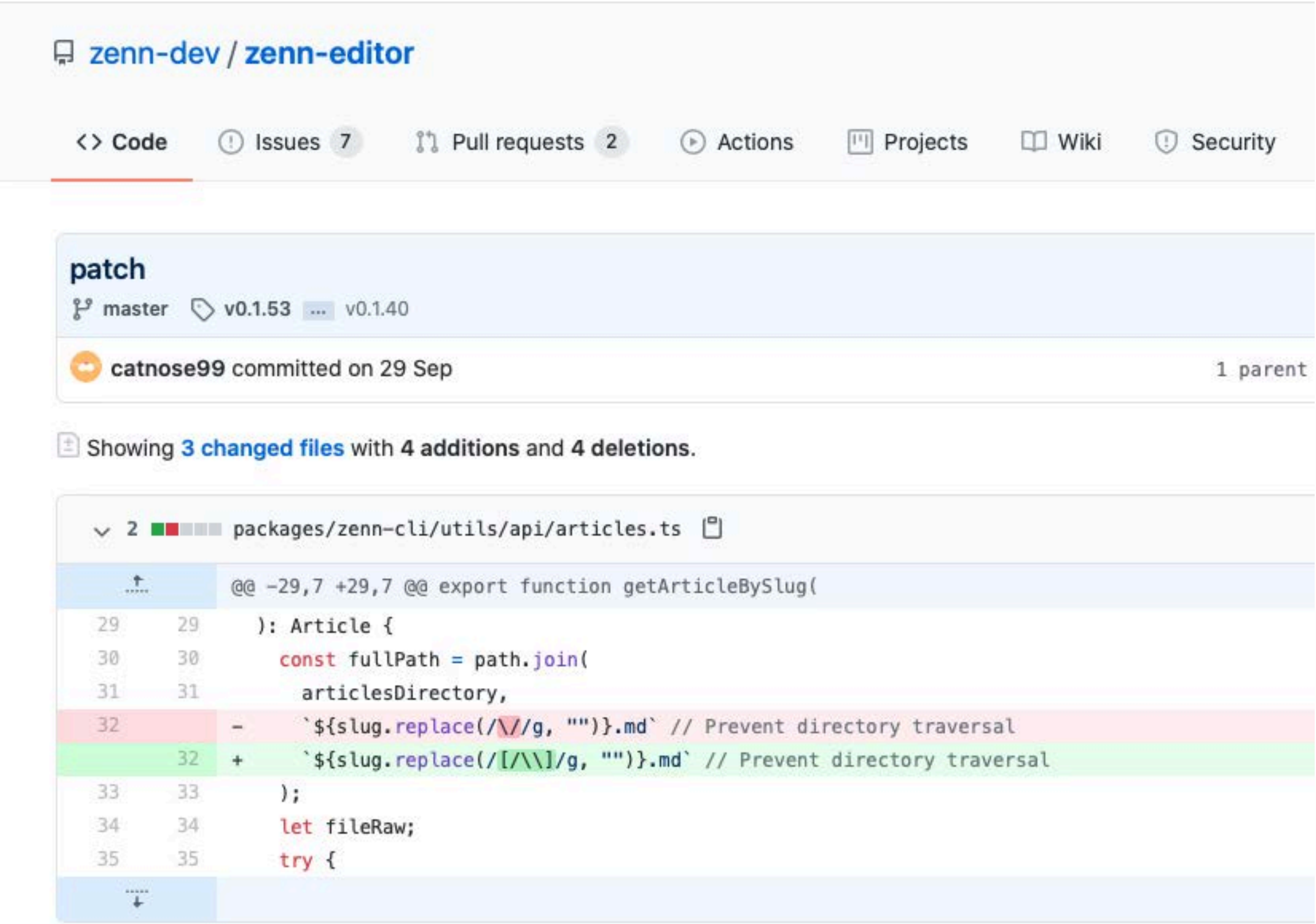
This addresses some of the suggestions raised by folks in #809.

CHTJonas added 2 commits on 27 Jul 2019

```
Add vulnerability warning to README ...    Verified  ✕ 0740ac9
Improve readability                        Verified  ✕ d68adb6
```

Examples of data we found useful

- Directory Traversal
- zenn-cli
- GitHub Description ”patch”
- Attempt at security by obscurity
- Regular Expression used was improper at sanitizing file paths



patch

master v0.1.53 ... v0.1.40

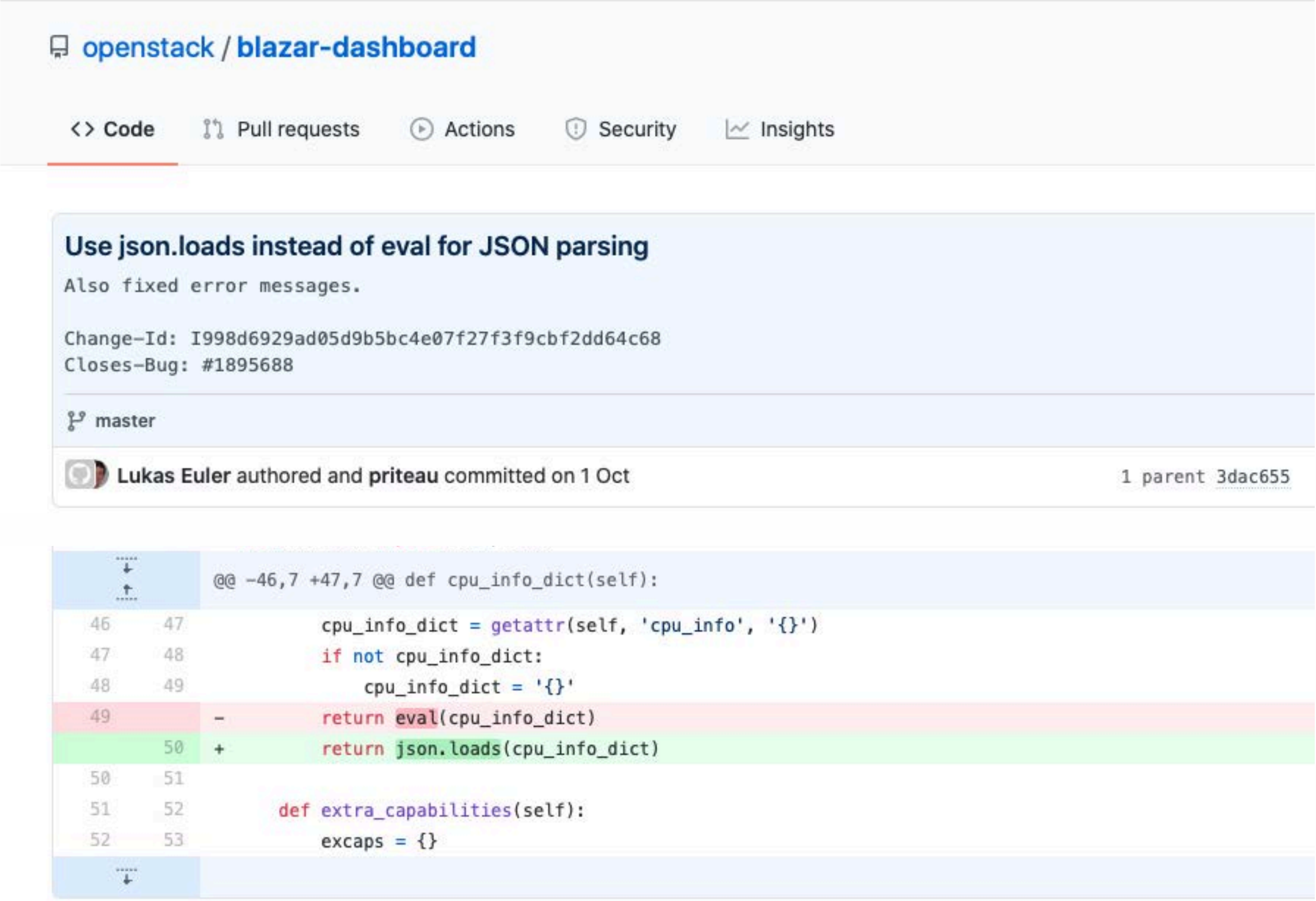
catnose99 committed on 29 Sep 1 parent

Showing 3 changed files with 4 additions and 4 deletions.

```
2 packages/zenn-cli/utils/api/articles.ts
@@ -29,7 +29,7 @@ export function getArticleBySlug(
   ): Article {
     const fullPath = path.join(
       articlesDirectory,
-      `${slug.replace(/\//g, "")}.md` // Prevent directory traversal
+      `${slug.replace(/[/\\]/g, "")}.md` // Prevent directory traversal
     );
     let fileRaw;
     try {
```

Examples of data we found useful

- Arbitrary Code Execution
- Unsafe use of eval during JSON parsing
- blazar_dashboard



Use json.loads instead of eval for JSON parsing

```
Also fixed error messages.

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

Lukas Euler authored and priteau committed on 1 Oct

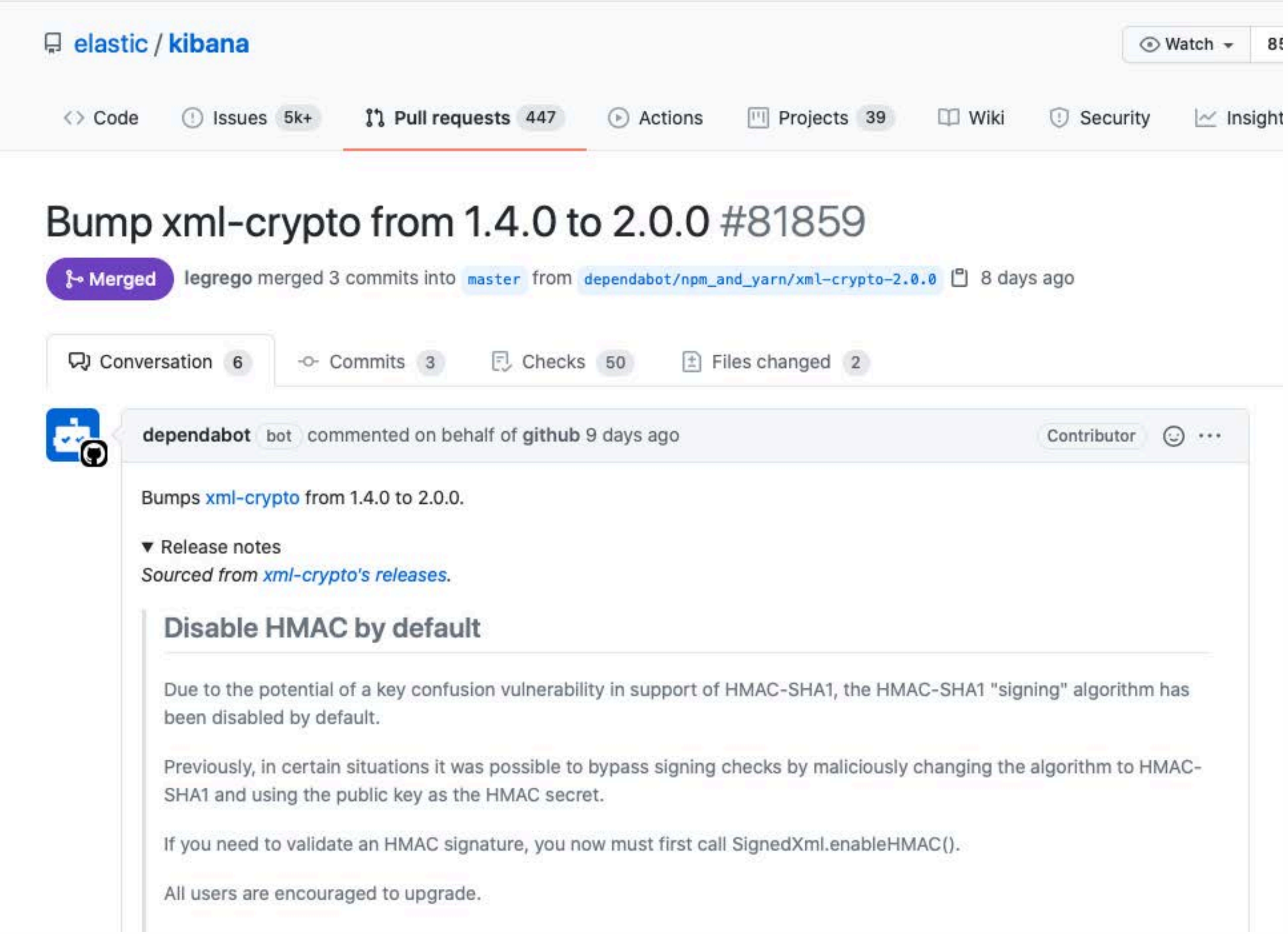
1 parent 3dac655

```
@@ -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)

      def extra_capabilities(self):
          excaps = {}
```

Examples of data we found useful

- Transitive Vulnerability
- Key Confusion
- Vulnerable xml-crypto is transitively found in kibana
- >15k stars



Bump xml-crypto from 1.4.0 to 2.0.0 #81859

Merged legrego merged 3 commits into `master` from `dependabot/npm_and_yarn/xml-crypto-2.0.0` 8 days ago

dependabot bot commented on behalf of github 9 days ago

Bumps xml-crypto from 1.4.0 to 2.0.0.

▼ Release notes

Sourced from xml-crypto's releases.

> ### Disable HMAC by default
>
> Due to the potential of a key confusion vulnerability in support of HMAC-SHA1, the HMAC-SHA1 "signing" algorithm has been disabled by default.
>
> Previously, in certain situations it was possible to bypass signing checks by maliciously changing the algorithm to HMAC-SHA1 and using the public key as the HMAC secret.
>
> If you need to validate an HMAC signature, you now must first call SignedXml.enableHMAC().
>
> All users are encouraged to upgrade.

Key observations from processing the data

- Developers Fixing Code flaws/Contributors reporting security issues
  - Cross-site Scripting, Arbitrary Code Execution, Deserialization flaws, Directory Traversal, Denial of Service, Man-in-the-Middle, etc
- Developers Fixing Transitive issues
  - Developers updating outdated and/or vulnerable libraries
- Able to discover and act on issues faster

It really is worse than it looks

- So far we have talked about discovering known vulnerability
- New type(s) of vulnerabilities uncovered over time
  - Deserialization vulnerabilities (eg. jackson-databind, Ruby’s YAML)
  - Arbitrary File Overwrite/Directory Traversal through Zip/Unzip functions (eg. adm-zip, mholt/archiver, Apache Karaf, plexus-archiver)
  - Prototype Pollution (eg. merge >2m weekly downloads; at least 1760 dependents)
- Number of libraries expected to increase over time
  - Very probable to have at least a component that may be vulnerable in the future, even if they are safe today

Discovering similar vulnerabilities

- Collected baseline information on existing vulnerabilities
- Zoom into the pattern of each vulnerability type
  - Deserialization/Arbitrary Code Execution
  - Directory Traversal through Zip/Unzip
  - Prototype Pollution
- For each pattern, devise key signatures that can be used to automatically single out potential libraries
  - Automatically create Proof-of-Concept and get results
  - Akin to running a Dynamic Analysis
  - May not be possible to achieve for all types of vulnerabilities

Key Takeaways

- Increasingly challenging to keep up with the increase in the amount of open-source libraries, the usage that follows, and possibly its inherent vulnerabilities.
- Machine learning is efficient in narrowing down vulnerability related data, however the process is not self sufficient, and can still be improved upon.
- Vulnerabilities discussed are merely a subset of actual vulnerabilities, and we should at least be on par with what’s been found and disclosed, while trying to discover more vulnerabilities.

Questions?

- asharma@veracode.com
- mang@veracode.com
- www.veracode.com

Thank You