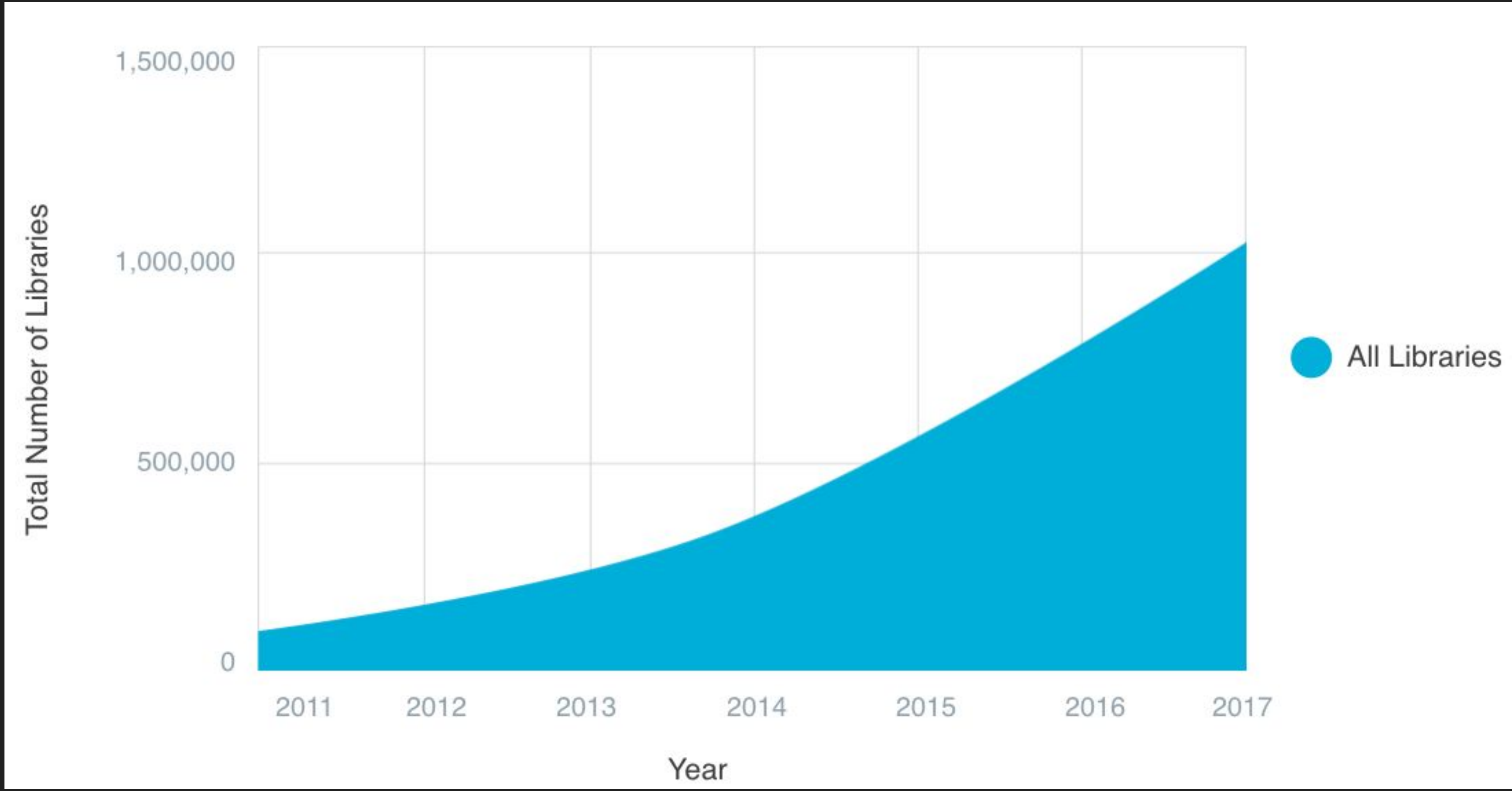
Using Machine Learning to Identify Security Issues in Open-Source Libraries

Asankhaya Sharma
Director, Software Engineering
CA Veracode

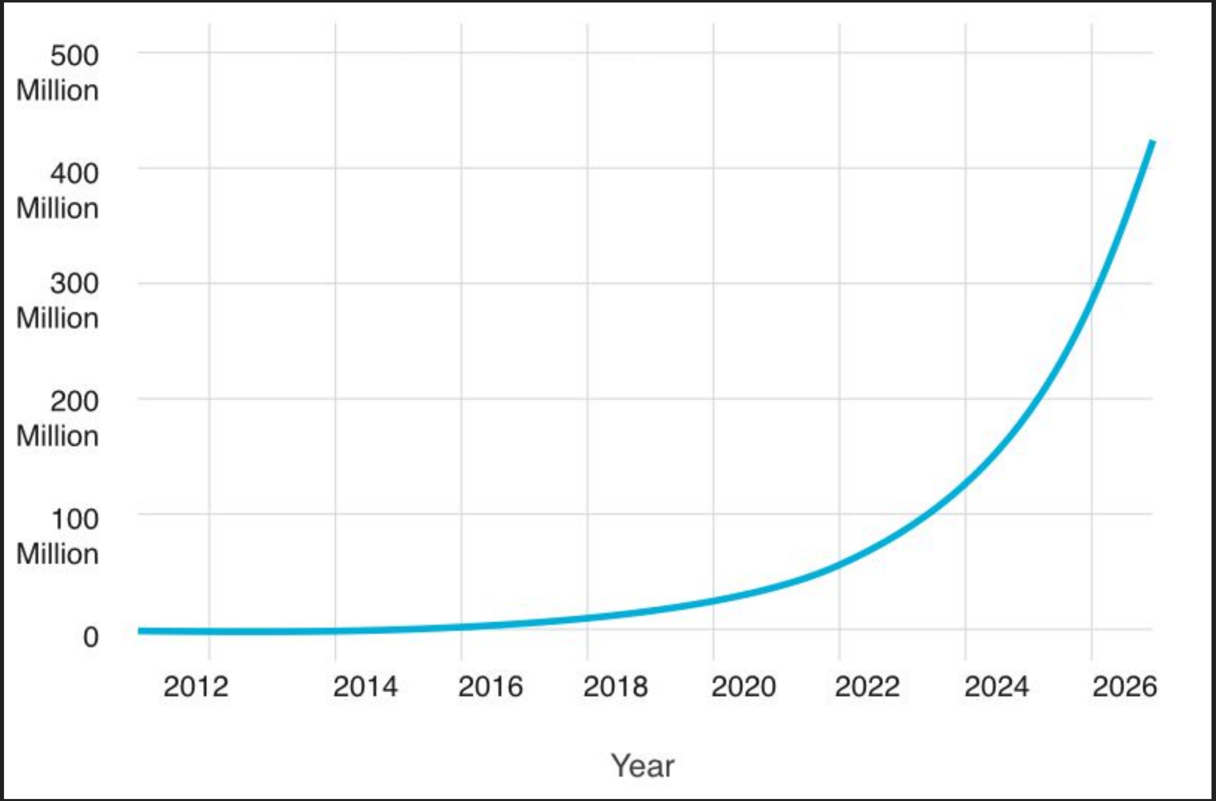
Outline

- Overview of the problem space
- Unidentified security issues
- How Machine Learning can help
- Machine Learning at Veracode
- Results

Open-Source Library Growth

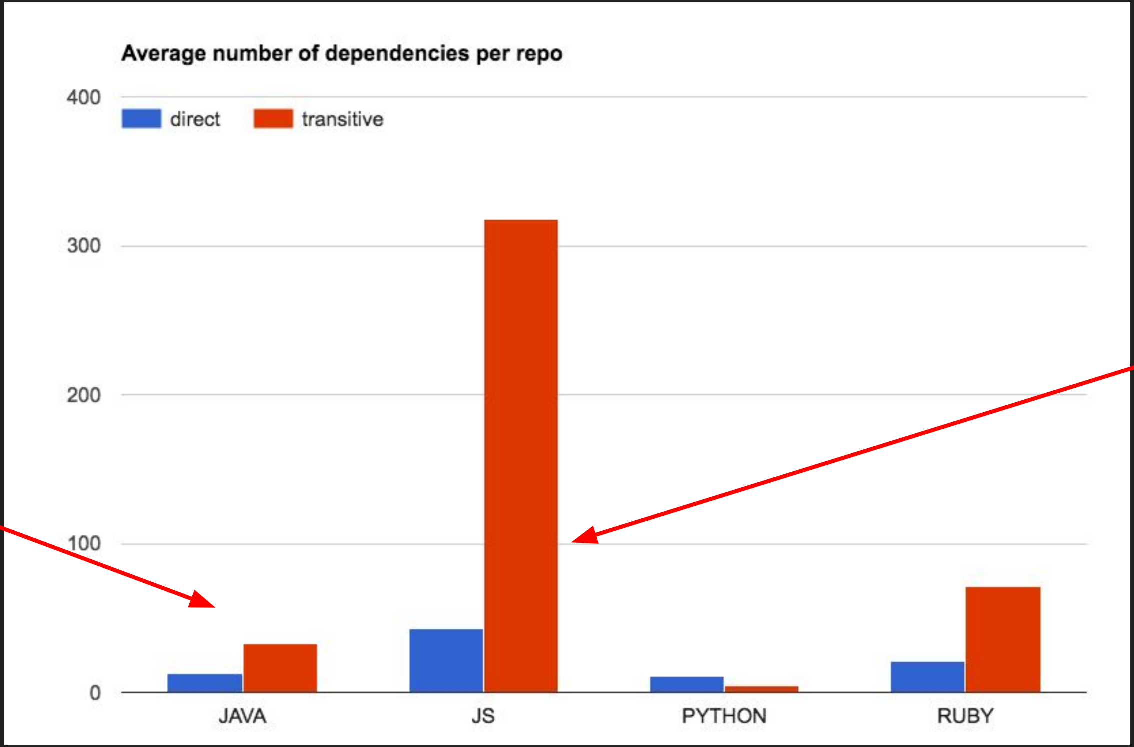


Projection: > 400M Libraries by 2026



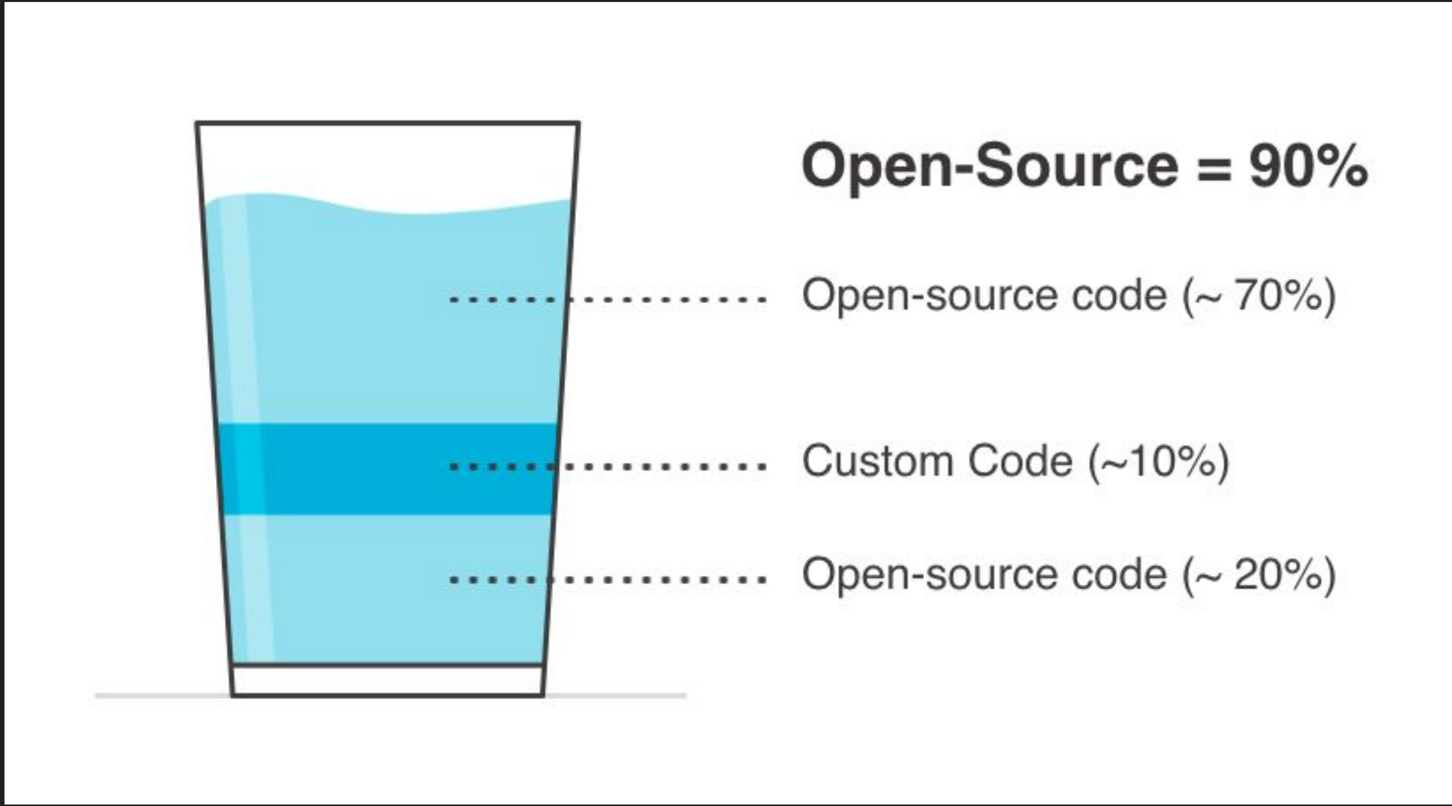
Complexity of Libraries has exploded

For every 1 Java library you add to your projects, 4 others are added



For every one library you add to a Node.js project, 9 others are added

The Code Cocktail

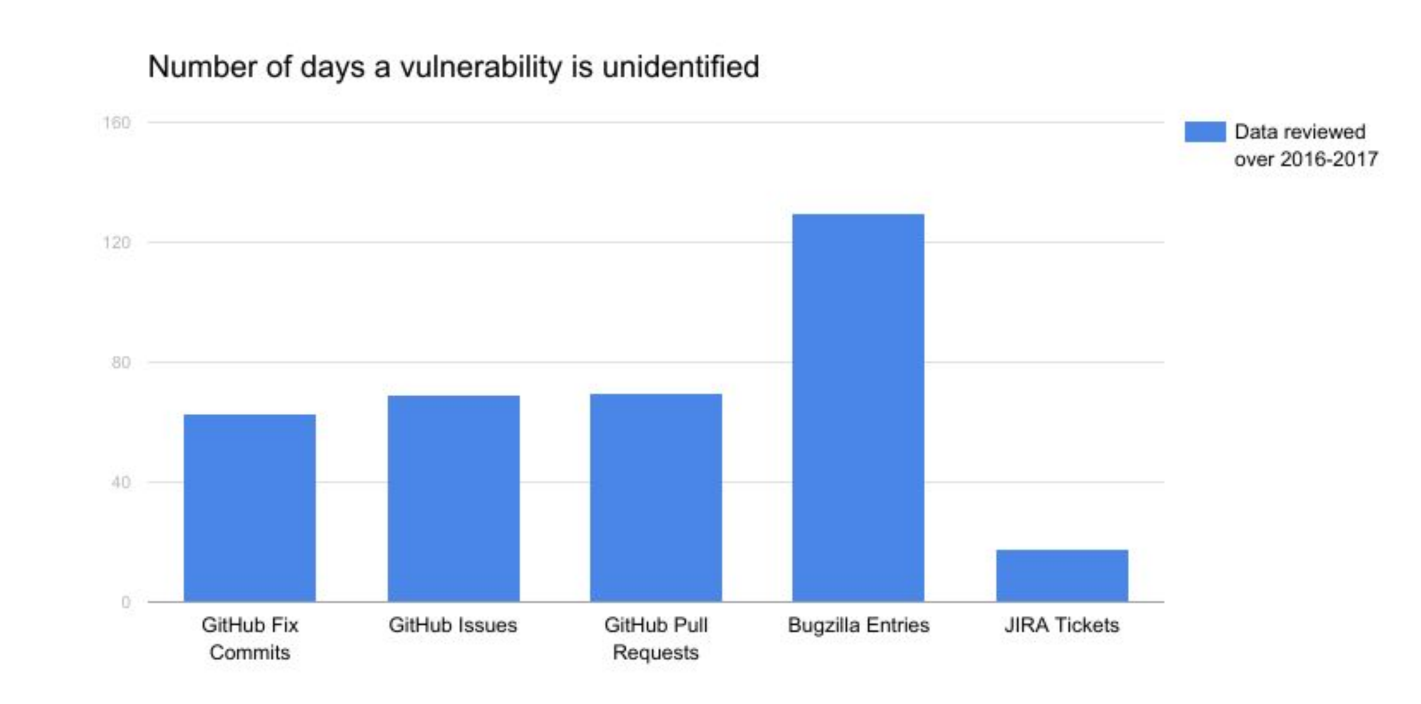


Vulnerabilities in Open-Source Libraries

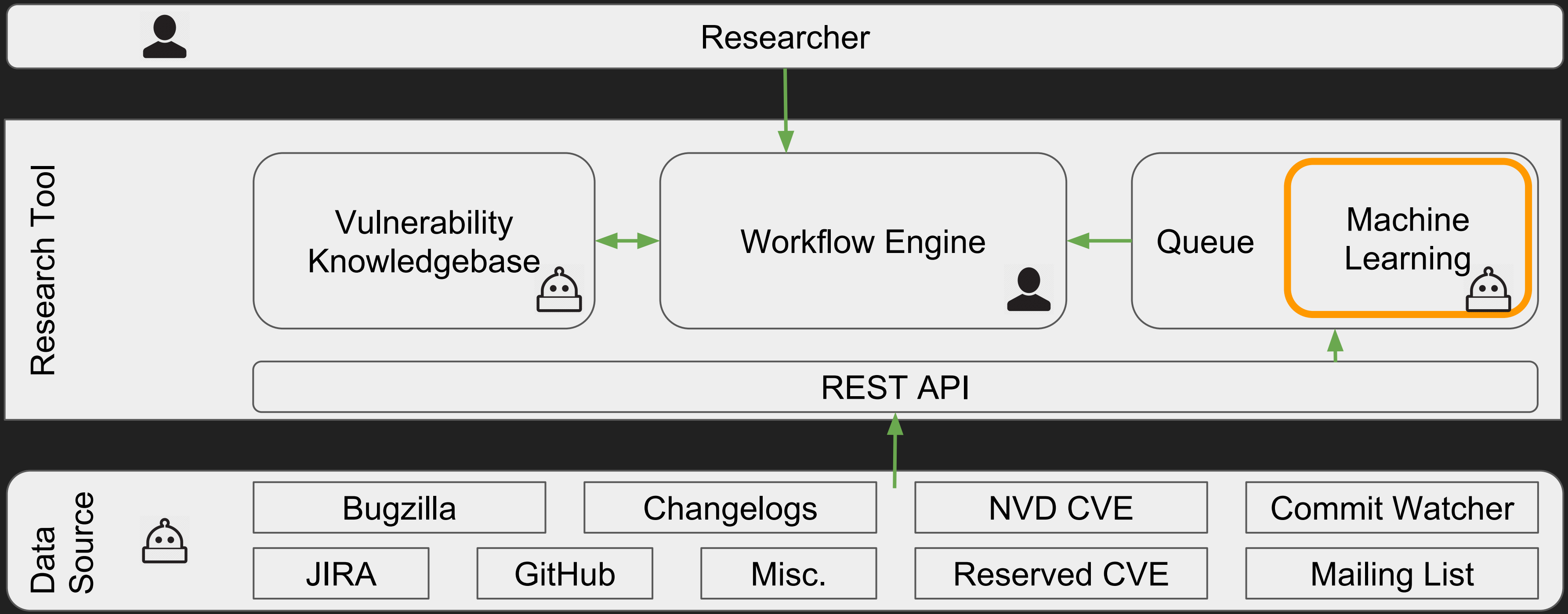
- Known Sources
  - CVEs / NVD
  - Advisories
  - Mailing list disclosures
- Unidentified issues
  - Commit logs
  - Bug reports
  - Change logs
  - Pull Requests

Security Issues are often not reported or publicly mentioned

Mining for unidentified vulnerabilities



WOPR: Tool for Reviewing Unidentified Issues

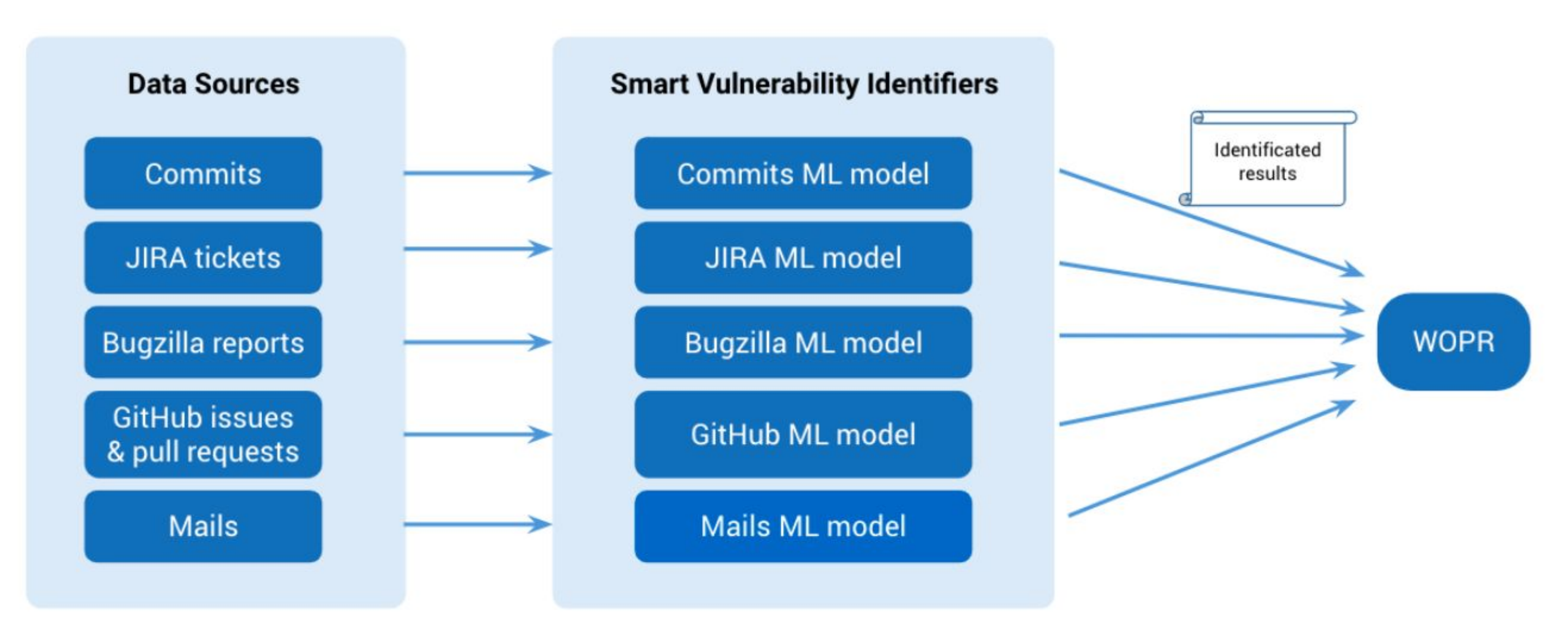


Machine Learning for Identifying Vulnerabilities

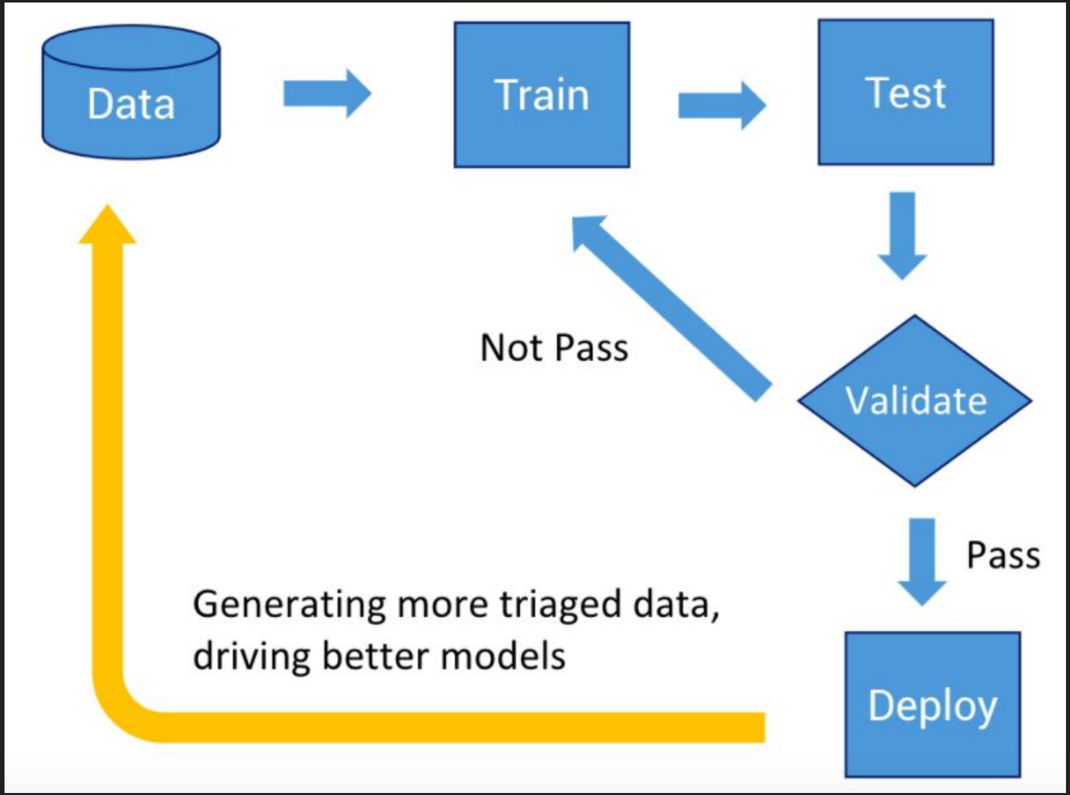
“do machine learning like the great engineer you are, not like the great machine learning expert you aren’t.”

Martin Zinkevich, Rules of Machine Learning: Best Practices for ML Engineering
http://martin.zinkevich.org/rules_of_ml/rules_of_ml.pdf

System overview



ML Pipeline


Data
Train
Test
Not Pass
Validate
Pass
Generating more triaged data,
driving better models
Deploy

Data collection

- Regular expression to filter out security-unrelated issues
  - Rule sets cover almost all possible expressions related to security issues
- Tracked ~6000 projects in 6 languages
  - Tracked languages: Java, Python, Ruby, JavaScript, Objective C, and Go
- Ground truth datasets
  - Professional security researchers label all data, and create vulnerability reports
  - Available at SourceClear Vulnerability Database

Source	# of tracked projects
Github	2070
JIRA	1310
Bugzilla	2224

Datasets

Highly imbalanced

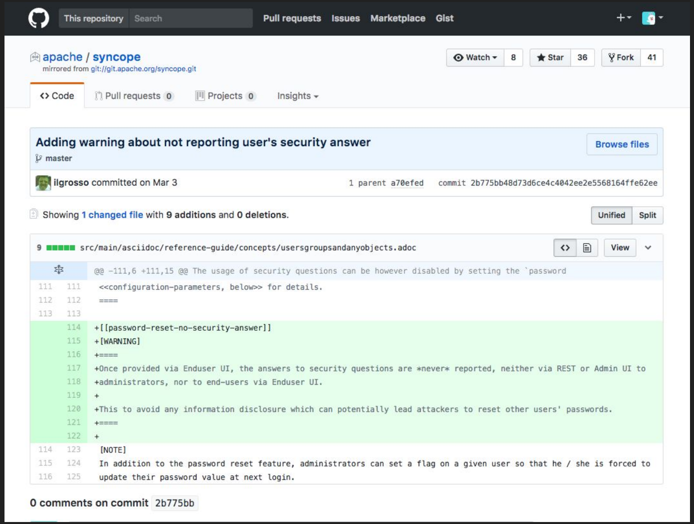
Dataset	Size	# vulnerability_related	Imbalanced ratio
Commit	12409	1303	10.50%
GitHub bug reports	10414	612	5.88%
JIRA bug reports	11145	204	1.83%
Bugzilla bug reports	2629	1089	41.42%
Mails	4499	2721	60.48%

Commits & bug reports initial training data: Jan. 2012 - Feb. 2017
Mails initial training data: Feb. 2017 - Aug. 2017

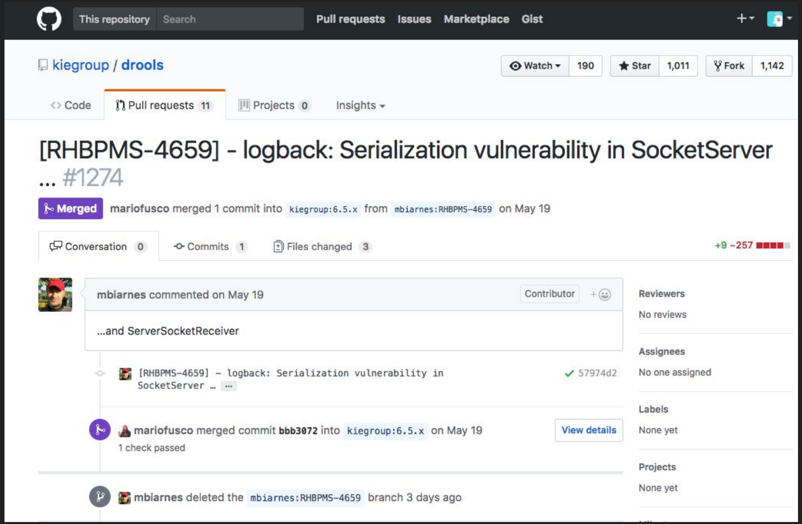
Samples

Noisy, diverse, mixed with urls, directories, variable names...

Commit



Bug report



Features

Commits

- Commit messages
- Comments
  - Most null
- Project name
  - Might impact prediction on projects not in training data
- Name of author
  - Common names and changed names etc

Bug reports

- Title
- Description
- Comments, number of comments
- Number of attachments
- Labels
- Created date and Last edited date

Mails

- Subject
- Content
- Sender

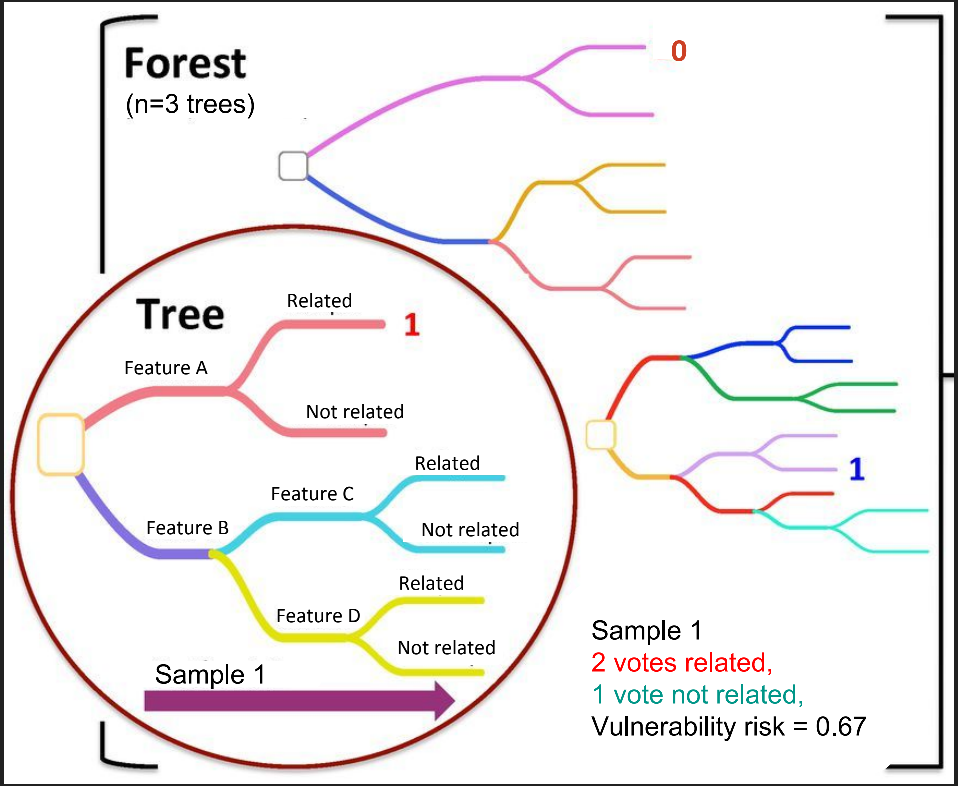
Text feature-Word embedding

- Word embedding
  - Map words to vectors so that computers can understand
- Word2vec
  - A word embedding method that uses a shallow 2-layer neural network to learn vector representation of words based on similarity in context

```
>>> word2vec['xss']
array([-0.06691808,  0.01889833,  0.08988539,  0.03727728,  0.09463213,
       0.04498576,  0.02401953,  0.01821383, -0.04510168,...., -0.00888534], dtype=float32)
>>> word2vec.most_similar('xss')
[(u'vulnerability', 0.6009132862091064), (u'attacks', 0.5554373860359192), (u'forgery',
0.4951219856739044), (u'spoofing', 0.49092593789100647), (u'dos', 0.4852156937122345), (u'prevention',
0.48259809613227844), (u'clickjacking', 0.48095956444740295), (u'protection', 0.46756529808044434),
(u'csrf', 0.457594096660614), (u'vuln', 0.4533842206001282)]
```

Built word2vec model based on 3 million unfiltered commits

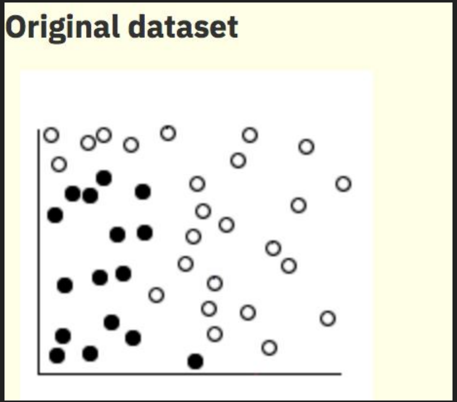
First training attempts-random forest

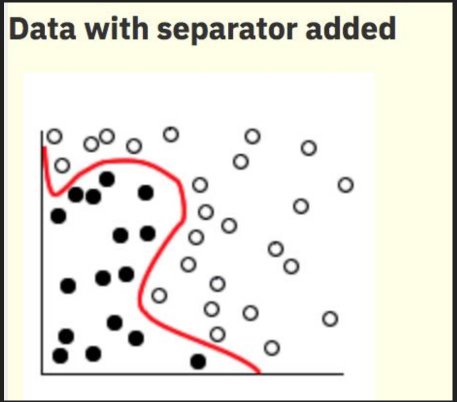


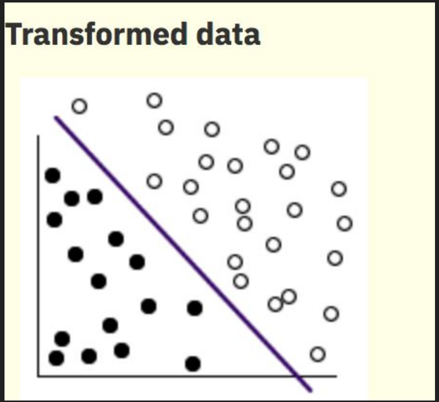
How Random Forest works?

- Training
  - Generate a forest of binary decision trees through randomly sampling a subset of train set and fitting
- Prediction
  - Each data sample traverses each tree until it reaches a leaf
  - At the leaf node, each tree creates a vote, the proportion of related votes is the prediction for the data sample

First training attempts-Support Vector Machine (SVM)





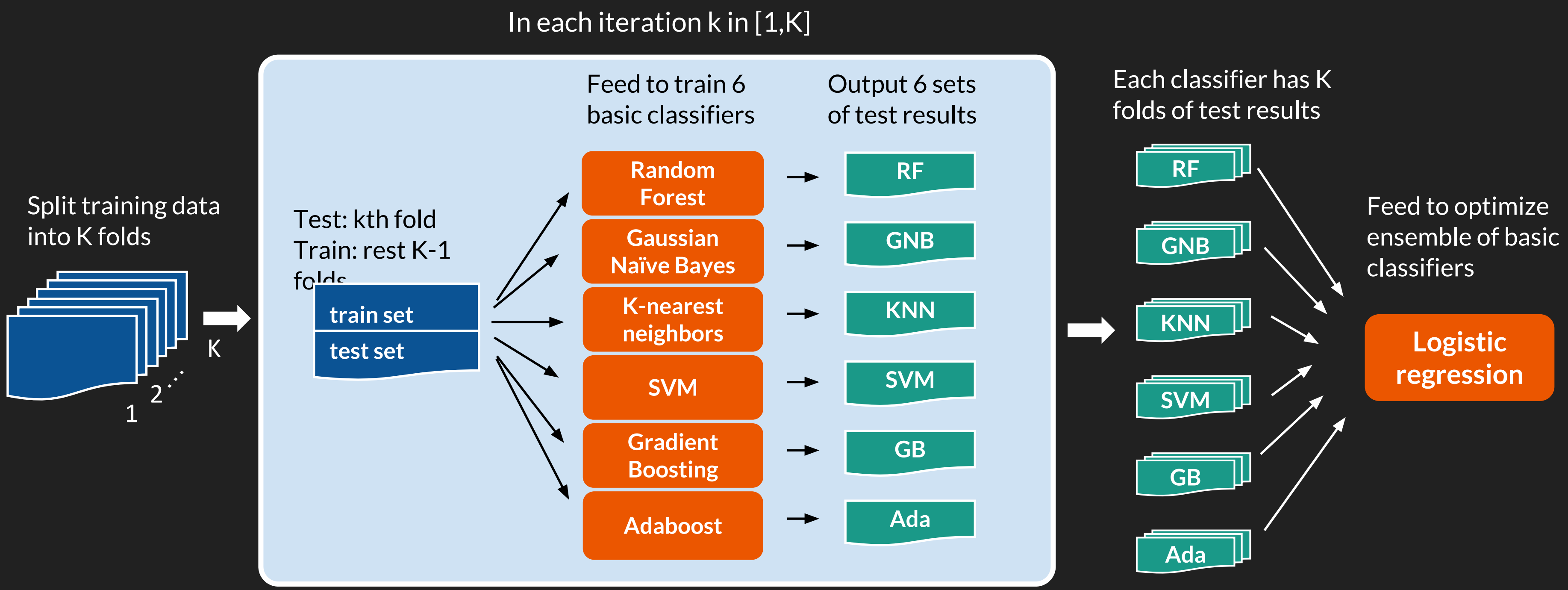


How SVM works?

- Mapping data to a high-dimensional feature space so that data points can be separated with a large ‘gap’ (or margin)
- Kernel - Mathematical function used for transformation
  - Linear
  - Polynomial
  - RBF (Radial basis function)

Unfortunately, these basic binary classifiers, even with best tuning parameters, failed us...

K-fold stacking



Evaluation-metrics

- **Precision rate**
  - Helps us focus on true vulnerabilities and save manual work on false positives

$$Precision = \frac{true\ positive}{true\ positive + false\ positive}$$

- **Recall rate**
  - Indicates the coverage of existing vulnerabilities

$$Recall\ rate = \frac{true\ positive}{true\ positive + false\ negative}$$

- **Probability threshold of vulnerability to control the tradeoff between two metrics**

Predicted positives

Commits (Total)	Commits (Positive)	Commits (Negative)	True positive	False positive
1000	100	900	70	35

Totally (70+35) = 105 shown to researchers

- Precision rate = 70/ (70+35) = 66.67%
- Recall rate = 70/ 100 = 70%
- Filtered commits: 895, 89.5%

Evaluation-test results of commits

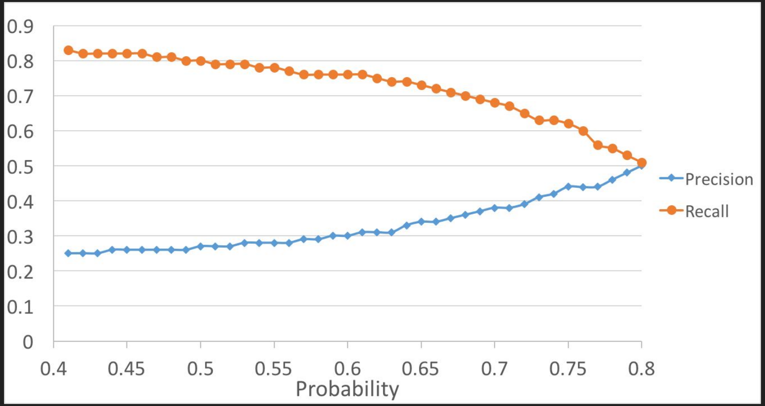


Figure: Identification performance of our stacking approach under commits

Table: Comparison with basic classifiers under the same recall rate in commits

Classifier	Recall rate	Precision (compared classifier vs.stacking)
Linear SVM	0.72	0.22 vs. 0.34
Logistic Regression	0.76	0.22 vs. 0.31
Random Forest	0.76	0.19 vs. 0.31
Gaussian Naive Bayes	0.77	0.14 vs. 0.28

Details in the paper

ACM DL DIGITAL LIBRARY

Automated identification of security issues from commit messages and bug reports

Full Text: PDF Get this Article

Authors: Yaqin Zhou SourceClear, Singapore
Asankhaya Sharma SourceClear, Singapore

2017 Article

Published in:

- Proceeding

ESEC/FSE 2017 Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering
Pages 914-919

Paderborn, Germany — September 04 - 08, 2017
ACM New York, NY, USA ©2017
table of contents ISBN: 978-1-4503-5105-8
doi>10.1145/3106237.3117771

Bibliometrics

- Citation Count: 0
- Downloads (cumulative): 203
- Downloads (12 Months): 203
- Downloads (6 Weeks): 15

Link to PDF

Production observation

- The initial 3-months observation from commits
  - Deployed Model
    - 12-fold stacking with probability threshold 0.75
    - Test precision 0.44 and recall rate 0.62
  - Added ~3000 new projects
    - 2070 -> 5002
  - Precision **0.83** and recall rate **0.74**

Commits (Total)	Commits (Positive)	Commits (Negative)	True positive	False positive
2268	215	2053	160	32

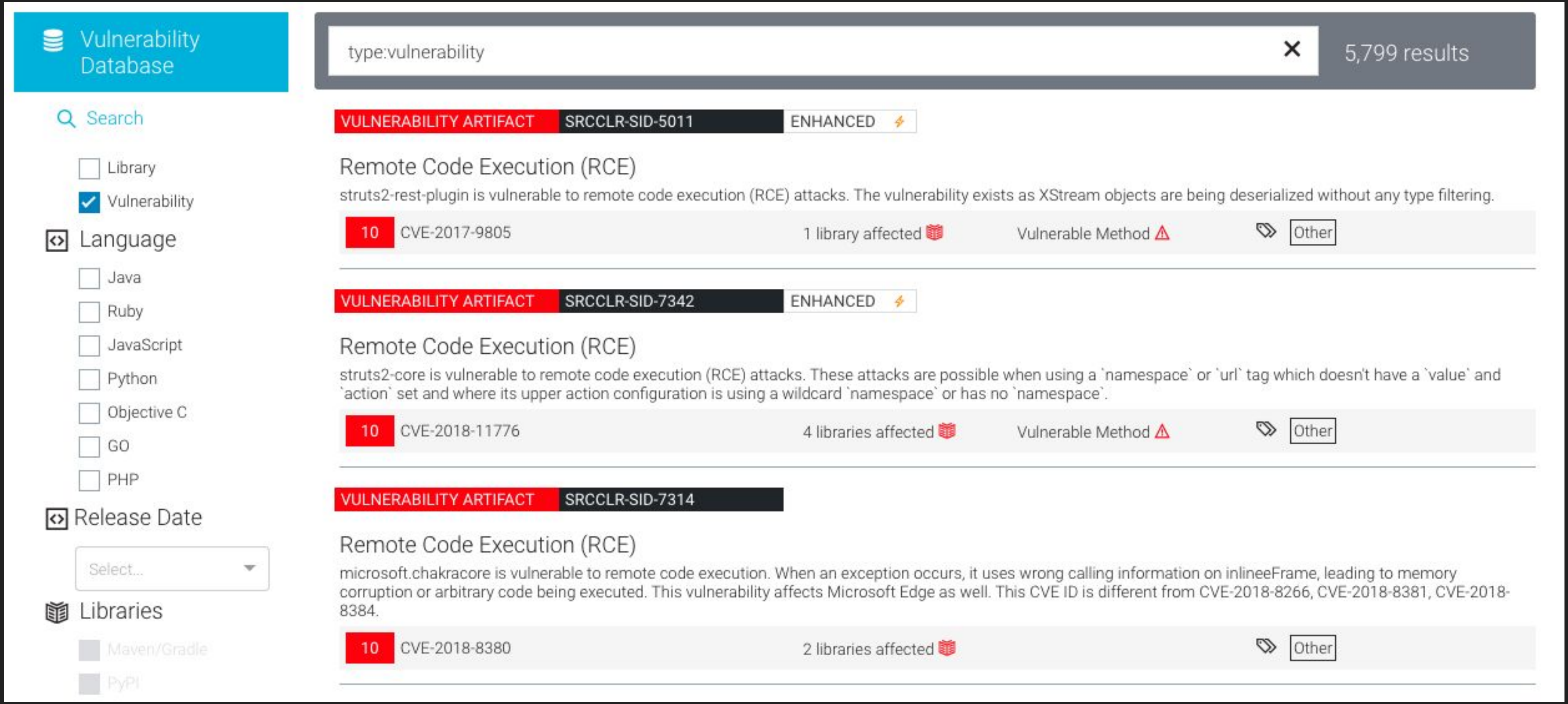
Production observation

- Track vulnerabilities at large scale and low cost in real time
  - Increased number of projects, e.g., for Github, 6 times more

Sources	GitHub	JIRA	Bugzilla
#Projects	11790	1310	2224

  - Accelerate vulnerability identification
    - 87 Go lang vulnerabilities identified from Commits
    - 33 Go lang vulnerabilities identified from Github Issues
- Current Github/Jira issues can spot vulnerabilities at the first time

Epilogue



Vulnerability Database
Search
Library
Vulnerability
Language
Java
Ruby
JavaScript
Python
Objective C
GO
PHP
Release Date
Select...
Libraries
Maven/Gradle
PyPI
type:vulnerability
5,799 results
VULNERABILITY ARTIFACT
SRCCLR-SID-5011
ENHANCED
Remote Code Execution (RCE)
struts2-rest-plugin is vulnerable to remote code execution (RCE) attacks. The vulnerability exists as XStream objects are being deserialized without any type filtering.
10
CVE-2017-9805
1 library affected
Vulnerable Method
Other
VULNERABILITY ARTIFACT
SRCCLR-SID-7342
ENHANCED
Remote Code Execution (RCE)
struts2-core is vulnerable to remote code execution (RCE) attacks. These attacks are possible when using a `namespace` or `url` tag which doesn't have a `value` and `action` set and where its upper action configuration is using a wildcard `namespace` or has no `namespace`.
10
CVE-2018-11776
4 libraries affected
Vulnerable Method
Other
VULNERABILITY ARTIFACT
SRCCLR-SID-7314
Remote Code Execution (RCE)
microsoft.chakracore is vulnerable to remote code execution. When an exception occurs, it uses wrong calling information on inlineeFrame, leading to memory corruption or arbitrary code being executed. This vulnerability affects Microsoft Edge as well. This CVE ID is different from CVE-2018-8266, CVE-2018-8381, CVE-2018-8384.
10
CVE-2018-8380
2 libraries affected
Other

Thanks!

We are actively hiring in Singapore for CA Veracode …

- Software Engineer
- Principal Software Engineer
- Security Program Manager
- Senior Support Engineer
- Associate Security Scan Specialist
- Static Operations Engineer

Apply now