



Detect complex code patterns using semantic grep

Colleen Dai | colleen@returntocorp.com

 [@r2cdev](https://twitter.com/r2cdev)

Slides will be posted

tl;dw – This Talk

- Secure code is hard
- Static analysis tools are too noisy / too slow
- grep isn't expressive enough
- Need something, fast, code-aware, flexible, powerful... **open source!**



[Semgrep](#): Fast and syntax-aware semantic code pattern search for many languages: like grep but for code

Semgrep

Use to:

- Search: Find security bugs
- Guard: Enforce specific patterns and best practices
- Migrate: Easily upgrade from deprecated APIs

who is?

me:

Colleen Dai, security software engineer @ r2c
Graduated Stanford with B.S. of C.S., M.S. Stats



r2c:


We're an SF based static analysis startup on a mission to profoundly improve software security and reliability.



Outline

1. Background

2. `grep` and Abstract Syntax Trees (ASTs)
3. Semgrep Examples!
4. Using Semgrep for Mass Scanning
5. Integration into CI/CD
6. Semgrep Rules Registry

 **returntocorp / semgrep**

Watch 35

Unstar 2k

Fork 77

<> Code

Issues 168

Pull requests 6

Actions

Security

...

develop







Go to file

Add file

Code

About



	emjin Update pattern-from-code ...	18 seconds ago	1,327
	.circleci	Use new python rule to detect wro...	17 days ago
	.github	add basic metrics for semgrep-co...	6 days ago
	.vscode	add pre-commit	8 months ago
	docs	release changes	2 days ago
	ocaml-tree-sit...	use latest ocaml-tree-sitter and nf...	7 days ago

Lightweight static analysis for many languages. Find bug variants with patterns that look like source code.

 semgrep.dev

static-analysis

github.com/returntocorp/semgrep

Semgrep, Est. 2009



First version of Semgrep (sgrep/pfff) was written at Facebook circa 2009 and was used to enforce nearly 1000 rules!

The original author, Yoann Padialeau ([@aryx](#)), joined r2c last year. Yoan was the first static analysis hire at Facebook and previously PhD @ Inria, contributor to coccinelle.lip6.fr

Language Support

Semgrep works for:



JSON



Python



OCaml



Ruby

Branch: **develop** ▾

semgrep / LICENSE



returntocorp/semgrep is licensed under the
GNU Lesser General Public License v2.1

Primarily used for software libraries, the GNU LGPL requires that derived works be licensed under the same license, but works that only link to it do not fall under this restriction. There are two commonly used versions of the GNU LGPL.

Permissions

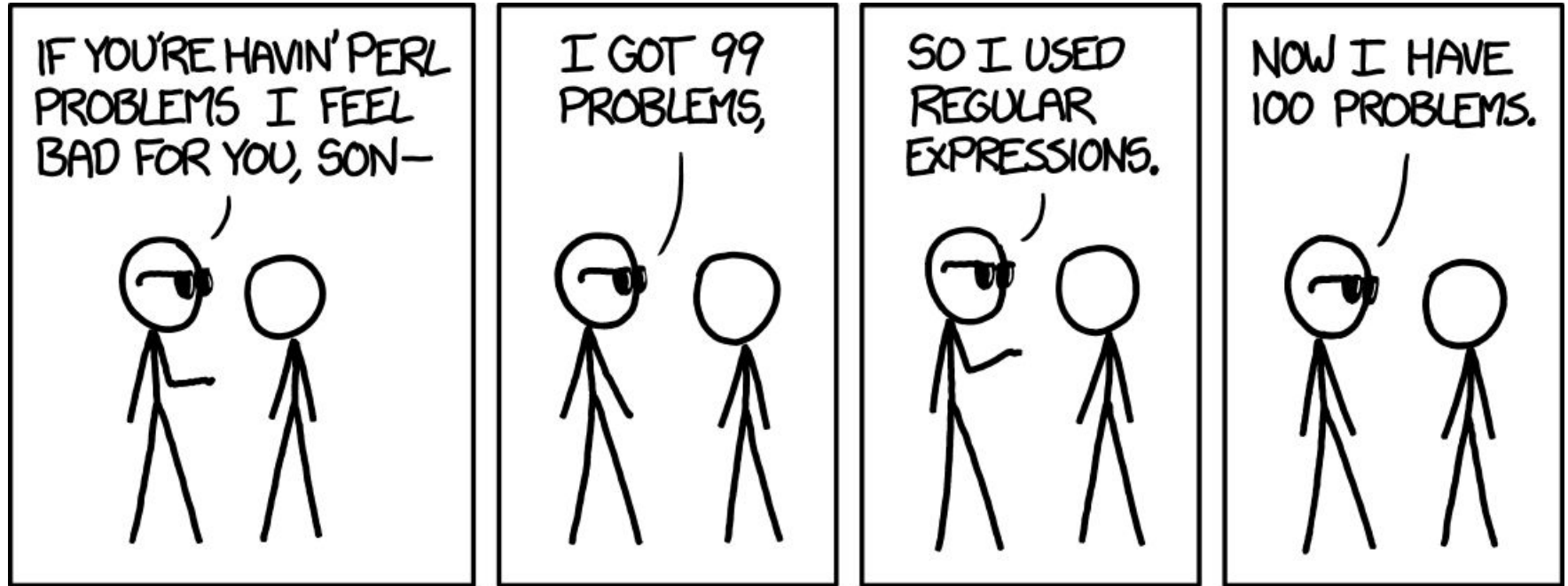
- ✓ Commercial use
- ✓ Modification
- ✓ Distribution
- ✓ Private use

This is not legal advice. [Learn more about repository licenses.](#)

Outline

1. Background
2. **grep and Abstract Syntax Trees (ASTs)**
3. Semgrep Examples!
4. Using Semgrep for Mass Scanning
5. Integration into CI/CD
6. Semgrep Rules Registry

xkcd 1171



Code is not a string, it's a tree



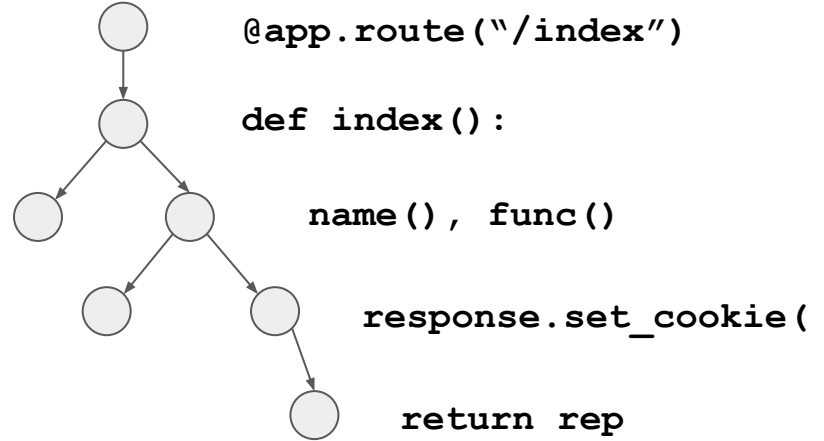
string

```
@app.route("/index")
def index():
    rep = response.set_cookie(name(),
    secure=False, s=func())
    return rep
```

!=



tree



Tree Matching

- Many tree matching tools: Gosec, Golint, Bandit, Dlint, ESLint, Flake8, Pylint, RuboCop, TSLint, and more!
- Have to become an **expert in every AST syntax** for every language your team uses
- Need **programming language expertise** to cover all idioms: languages have “more than one way to do it”
- **Commercial SAST tools?**
 - Complicated
 - Slow (not CI friendly)
 - Expensive

Find calls to `eval()`
in only 307 LOC 👍



```
yeonjuan Update: support globalThis (refs #12670) (#12774) 183e300 on Mar 17
20 contributors

307 lines (258 sloc) 9.24 KB Raw Blame History

1 /**
2  * @fileoverview Rule to flag use of eval() statement
3  * @author Nicholas C. Zakas
4  */
5
6 "use strict";
7
8 -----
9 // Requirements
10 -----
11
12 const astUtils = require("../utils/ast-utils");
13
14 -----
15 // Helpers
16 -----
17
18 const candidatesOfGlobalObject = Object.freeze([
19   "global",
20   "window",
21   "globalThis"
22 ]);
23
24 /**
25  * Checks a given node is a Identifier node of the specified name.
26  * @param {ASTNode} node A node to check.
27  * @param {string} name A name to check.
28  * @returns {boolean} `true` if the node is a Identifier node of the name.
29  */
30 function isIdentifier(node, name) {
31   return node.type === "Identifier" && node.name === name;
32 }
33
34 /**
35  * Checks a given node is a Literal node of the specified string value.
36  * @param {ASTNode} node A node to check.
37  * @param {string} name A name to check.
38  * @returns {boolean} `true` if the node is a Literal node of the name.
39  */
40 function isConstant(node, name) {
41   switch (node.type) {
42     case "Literal":
43       return node.value === name;
```

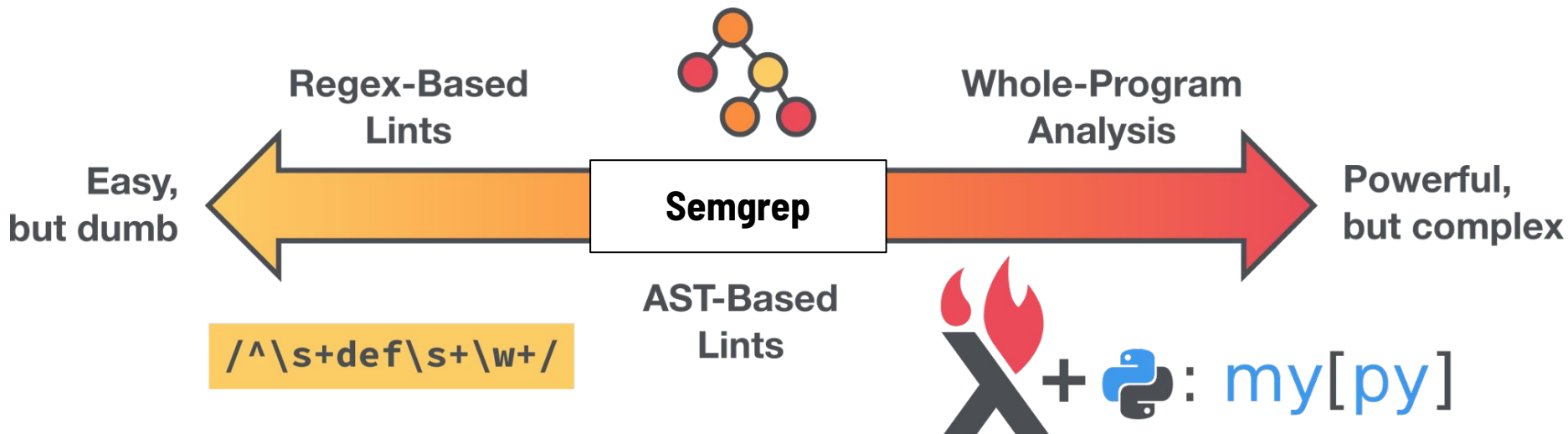
<https://github.com/eslint/eslint/blob/master/lib/rules/no-eval.js>

Static Analysis at Scale: An Instagram Story



Benjamin Woodruff [Follow](#)

Aug 15, 2019 · 13 min read



<https://instagram-engineering.com/static-analysis-at-scale-an-instagram-story-8f498ab71a0c>

Semgrep lets you reason about your **analysis**
the way you reason about your **code**.

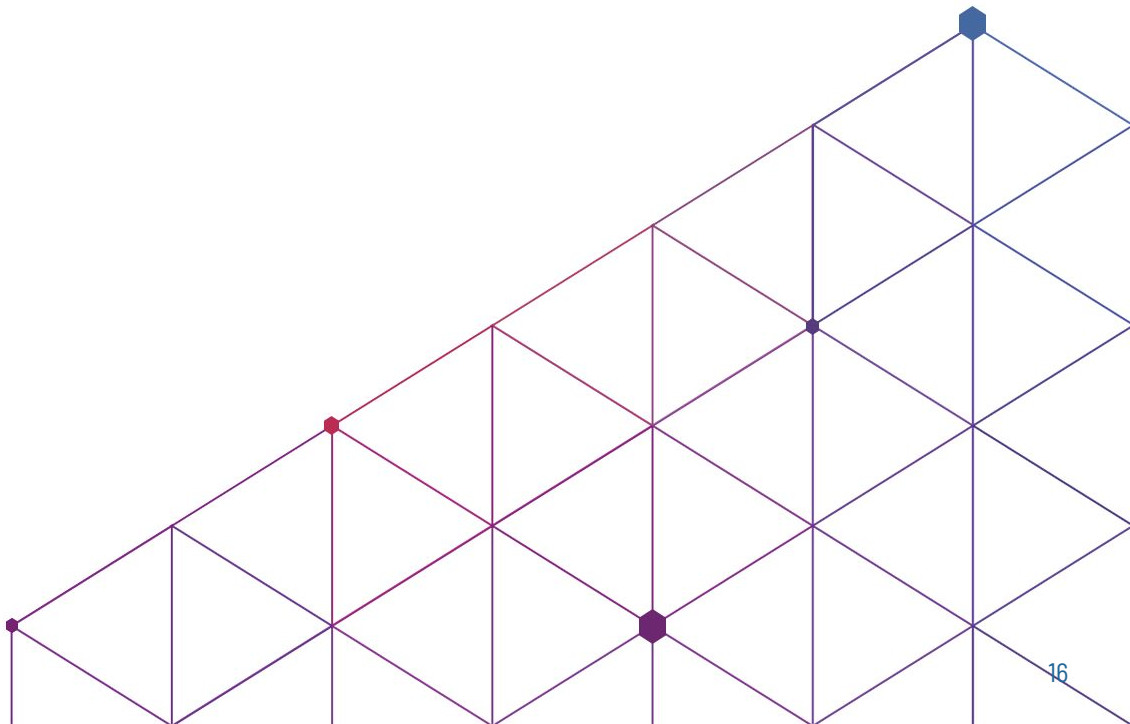
<https://r2c.dev/blog/2020/why-i-moved-to-semgrep-for-all-my-code-analysis/>

Outline

1. Background
2. `grep` and Abstract Syntax Trees (ASTs)
- 3. Semgrep Examples!**
4. Using Semgrep for Mass Scanning
5. Integration into CI/CD
6. Semgrep Rules Registry

Demos

1. Semgrep overview
 - a. Ellipsis ("...") operator
 - b. Metavariables
 - c. Composing patterns
2. Semgrep in Practice
 - a. Antipatterns
 - b. Business logic
3. Advanced Features
 - a. Extracting Routes
 - b. Autofix
 - c. Scripting



Finding Insecure Functions: Node Exec

(... operator)

```
exec("ls");
```

⇒ <https://semgrep.live/Xnw>

Full Solution: <https://semgrep.live/1Kk>

Finding Uses of `unsafe`

(Metavariables)

```
unsafe.Pointer(intPtr)  
unsafe.Sizeof(intArray[0])
```

⇒ <https://semgrep.live/nJNZ>

Full Solution: <https://semgrep.live/ZqLp>

Path Traversal with send_file

```
@app.route("/get_file/<filename>")
def get_file(filename):
    print("sending file", filename)
    return send_file(filename, as_attachment=True)
```

⇒ <https://semgrep.live/4bXx>

Full Solution: <https://semgrep.live/Pevp>

Cookies 🍪

```
@app.route("/index")  
def index():  
    r = response.set_cookie("username", "drew")  
    return r
```

⇒ <https://semgrep.live/8dJ>

Full Solution: <https://semgrep.live/vWX>

Configuration Files

This document describes `semgrep` configuration files and provides rule examples. Configuration files are specified with the `--config` (or `-f`) flag. A single [YAML](#) file or a directory of files ending in `.yaml` or `.yml` may be specified. Each configuration file must match the [schema](#).

For more information on the `--config` flag see [other configuration options](#).

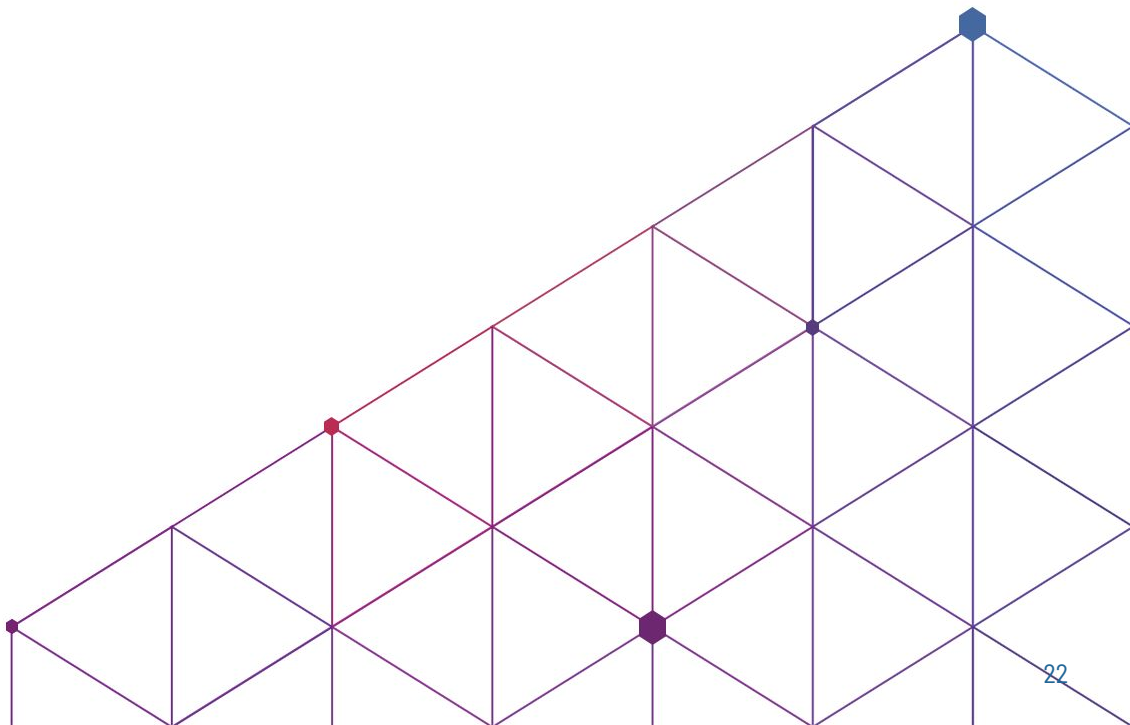
Contents:

- [Simple Example](#)
- [Other Configuration Options](#)
- [Schema](#)
- [Operators](#)
 - [pattern](#)
 - [patterns](#)
 - [pattern-either](#)
 - [pattern-regex](#)
 - [metavariable-regex](#)
 - [pattern-not](#)
 - [pattern-inside](#)
 - [pattern-not-inside](#)
 - [pattern-where-python](#)
- [Metavariable Matching](#)
 - [Metavariables in Logical ANDs](#)
 - [Metavariables in Logical ORs](#)
 - [Metavariables in Complex Logic](#)
- [Optional Fields](#)
 - [fix](#)
 - [metadata](#)
 - [paths](#)
- [Other Examples](#)
 - [Complete Useless Comparison](#)
- [Ignoring Findings](#)
- [Full Specification](#)

<https://github.com/returntocorp/semgrep/blob/develop/docs/configuration-files.md>

Demos

1. Semgrep overview
 - a. Ellipsis ("...") operator
 - b. Metavariables
 - c. Composing patterns
2. Semgrep in Practice
 - a. Antipatterns
 - b. Business logic
3. Advanced Features
 - a. Extracting Routes
 - b. Autofix
 - c. Scripting



Anti-pattern: Always True

```
if foo == foo:  
    print("foobar")  
else:  
    print("baz")
```

⇒ <https://semgrep.live/LkL>

Full Solution: <https://semgrep.live/scan?id=LkL&gitUrl=https://github.com/apache/libcloud>

Order of API Calls Must be Enforced

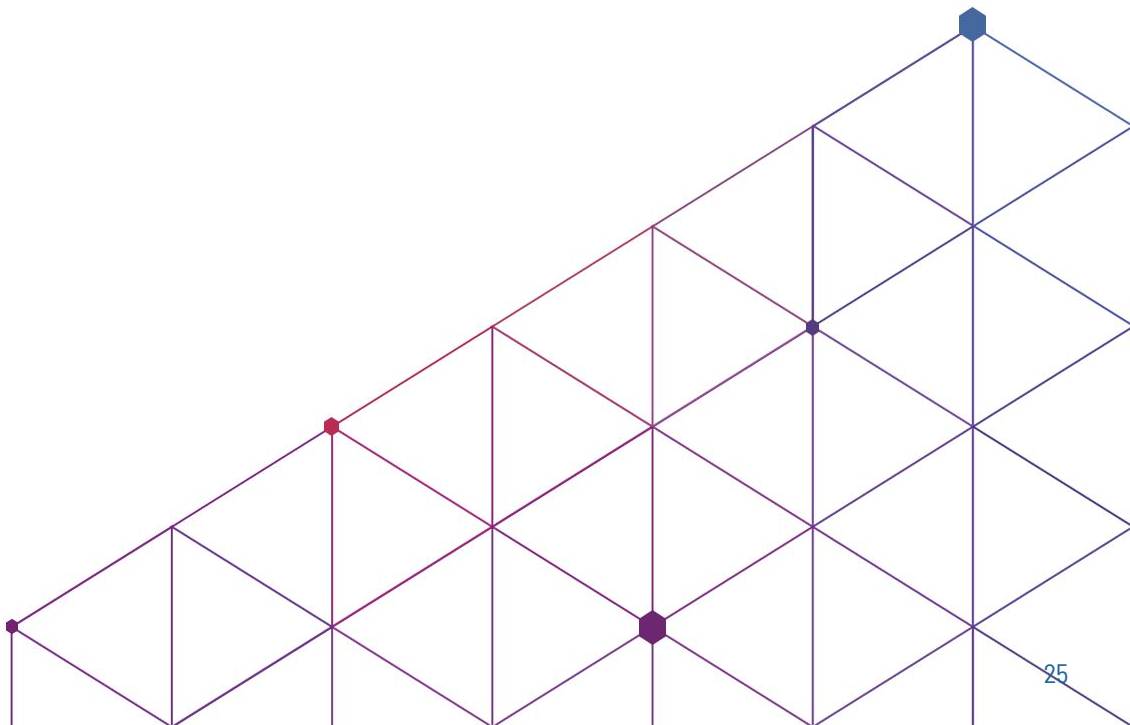
```
/*  
 * In this financial trading application, every transaction  
 * MUST be verified before it is made  
 *  
 * Specifically: verify_transaction() must be called on a transaction  
 * object before that object is passed to make_transaction()  
 */
```

⇒ <https://semgrep.live/6JqL>

Full Solution: <https://semgrep.live/oqZ6>

Demos

1. Semgrep overview
 - a. Ellipsis ("...") operator
 - b. Metavariables
 - c. Composing patterns
2. Semgrep in Practice
 - a. Antipatterns
 - b. Business logic
3. Advanced Features
 - a. Extracting Routes
 - b. Autofix
 - c. Scripting



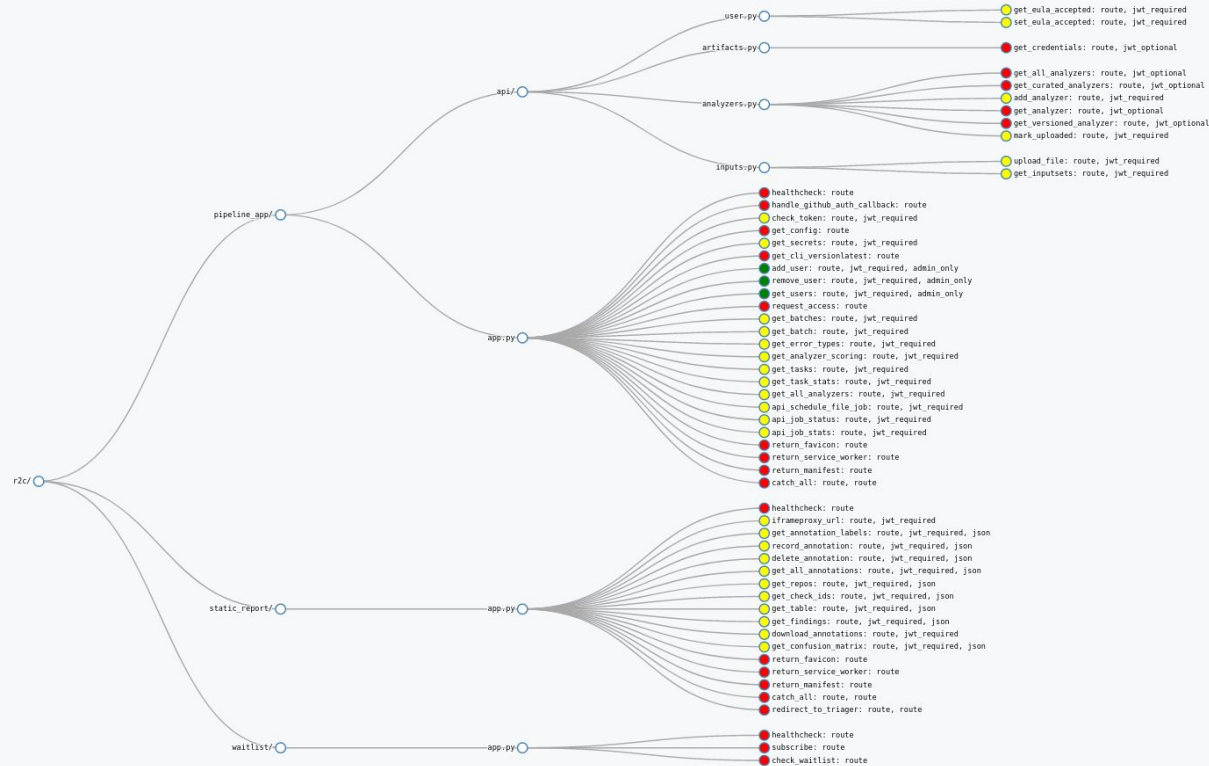
Know When New Routes Are Added

([Gorilla Toolkit](#))

```
func (a *App) initializeRoutes() {  
    a.Router.HandleFunc("/products",  
                        a.getProducts).Methods("GET")  
}
```

<https://semgrep.live/r6o1>

Semgrep application: code inventory



Autofix - Insecure SSL Configuration

```
&tls.Config{  
    KeyLogWriter: w,  
    MinVersion:  tls.VersionSSL30,  
    Rand:  randSource{}  
}
```

<https://semgrep.dev/xxyA/>

Scriptable (see docs)

your code here

```
rules:
  - id: use-decimalfield-for-money
    patterns:
      - pattern-inside: |
          class $M(...):
              ...
      - pattern: $F = django.db.models.FloatField(...)
      - pattern-where-python: 'price' in vars['$F']
      - message: "Found a FloatField used for variable $F. Use
        DecimalField for currency fields to avoid float-rounding errors."
    languages: [python]
    severity: ERROR
```

* requires non-default flag: **—dangerously-allow-arbitrary-code-execution-from-rules**

Use of Weak RSA Key

```
// Insufficient bit size  
pvk, err := rsa.GenerateKey(rand.Reader, 1024)  
  
// Sufficiently large bit size  
pvk, err := rsa.GenerateKey(rand.Reader, 2048)
```

⇒ <https://semgrep.live/zdRI>

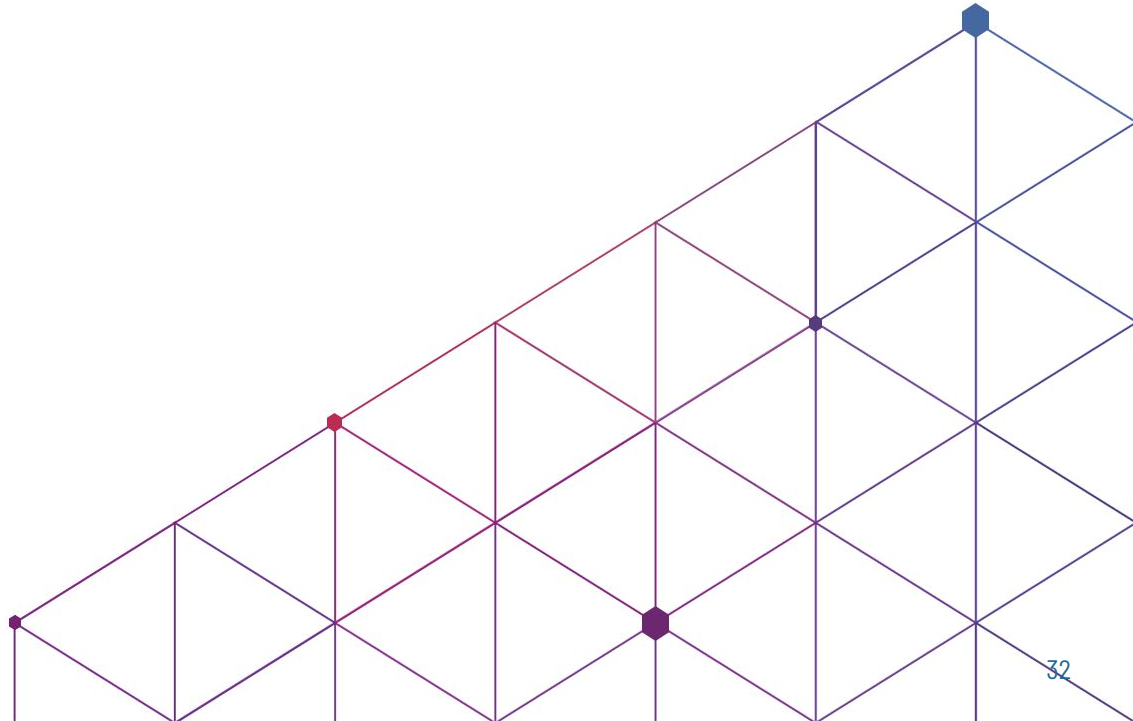
Full Solution: <https://semgrep.live/zdRI> | [docs](#)

Guard your code with secure defaults

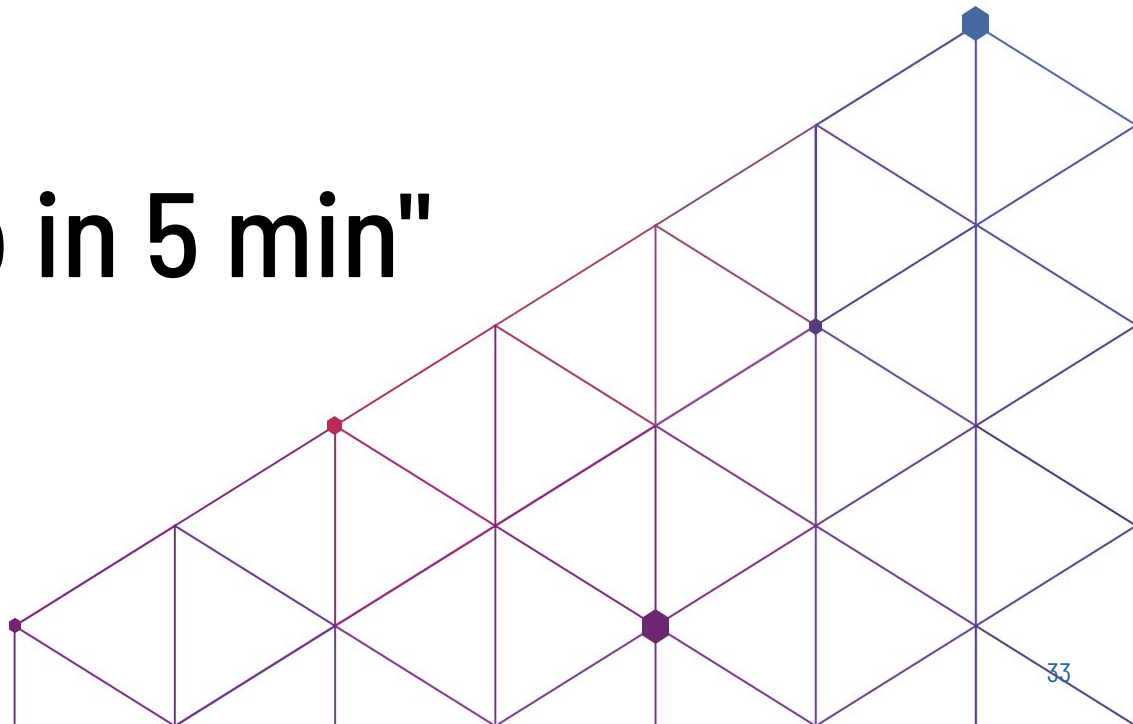
```
r = requests.post(url, verify=False)
```

<https://semgrep.live/editor?registry=python.requests.security.disabled-cert-validation>

break;



recap, a.k.a.
"learn semgrep in 5 min"



#1 Code equivalence (**semantic grep**)

```
$X == $X
```

Will match

```
(a+b != a+b) # <=> !(a+b==a+b)
```

```
foo(kwd1=1, kwd2=2, ...)
```

Will match

```
foo(kwd2=2, kwd1=1, kwd3=3)
```

```
subprocess.open(...)
```

Will match

```
from subprocess import open as  
sub_open  
  
result = sub_open("ls")
```

```
import foo.bar
```

Will match

```
from foo import bar
```

- **semgrep** knows about the semantics of the language, so one pattern can match variations of equivalent code (constant propagation! <https://semgrep.live/4K5>)

#2: '...' ellipsis operator

```
foo(...,5)
```

Will match

```
foo("...")
```

Will match

```
$V = get()  
...  
eval($V)
```

Will match

```
foo(1,2,3,4,5)  
foo(5)
```

```
foo("whatever sequence of chars")
```

```
user_data = get()  
print("do stuff")  
foobar()  
eval(user_data)
```

'...' can match sequences of:

- Arguments, parameters
- Characters
- Statements

#3 Metavariables (part 1)

```
foo($X,2)
```

Will match

```
foo(1,2)
```

```
if $E:  
    foo()
```

Will match

```
if x > 2:  
    foo()
```

```
if $X > $Y:  
    $S
```

Will match

```
if var > 2:  
    return 1
```

```
$F(1,2)
```

Will match

```
foo(1,2)
```

- **Metavariables** start with a \$ (\$X, \$Y, \$WHATEVER), contain uppercase ASCII characters
- **Matches:**
 - Expressions (including arguments)
 - Statements
 - Names (functions, fields, etc.)

#3 Metavariables (part 2)

```
$X == $X
```

Will match

```
if $E:  
    $S  
else:  
    $S
```

Will match

```
$V = open()  
close($V)
```

Will match

```
if (a+b == a+b) :
```

```
if x > 2:  
    foo()  
    bar()  
else:  
    foo()  
    bar()
```

```
myfile = open()  
close(myfile)
```

You can reuse the same metavariable: **semgrep** enforces **equality constraint**

Outline

1. Background
2. `grep` and Abstract Syntax Trees (ASTs)
3. Semgrep Examples!
- 4. Using Semgrep for Mass Scanning**
5. Integration into CI/CD
6. Semgrep Rules Registry

```
@$APP.route(...)  
def $FUNC(..., $FILENAME, ...):  
    ...  
    open(<... $FILENAME ...>, ...)
```

<https://semgrep.live/2Zz5/>

Start a job

Jobs

1131 dev/semgrep

0.14.0

1130 dev/semgrep

0.14.0

1129 dev/semgrep

0.14.0

1128 dev/semgrep

0.14.0

1127 dev/semgrep

0.14.0

github-1200-depends-on-flask(0.0.1)

yamlurl:https://semgrep.live/c/Kx5d

48 minutes ago

23.98% error rate

1126 dev/semgrep

0.14.0

github-1200-depends-on-flask(0.0.1)

yamlurl:https://semgrep.live/c/Kx5d

54 minutes ago

71.31% error rate

1125 dev/semgrep

0.14.0

github-1200-depends-on-flask(0.0.1)

yamlurl:https://semgrep.live/c/Kx5d

59 minutes ago

96.50% error rate

1124 dev/semgrep

0.14.0

npm-github-1000-latest-2019-09-17/...

yamlurl:https://semgrep.live/c/colleen...

4 days ago

2.40% error rate

1123 dev/semgrep

0.14.0

npm-latest-2019-08-26/0.0.2

yamlurl:https://semgrep.live/c/colleen...

4 days ago

0.00% error rate

Analyzer

dev/semgrep

0.14.0

Change

Step 3:
Select your parameters (optional)

Step 2:
Select your input set

Select input set

flask

depends-on-flask(0.0.1) 39.9k

github-1200-depends-on-flask(0.0.1) 1.2k

github-flask-talisman(0.0.1) 37

top1k-flask-github(0.0.1) 1k

Run Job

Filter by repositories, commit hashes, or checks:

filter by repos...

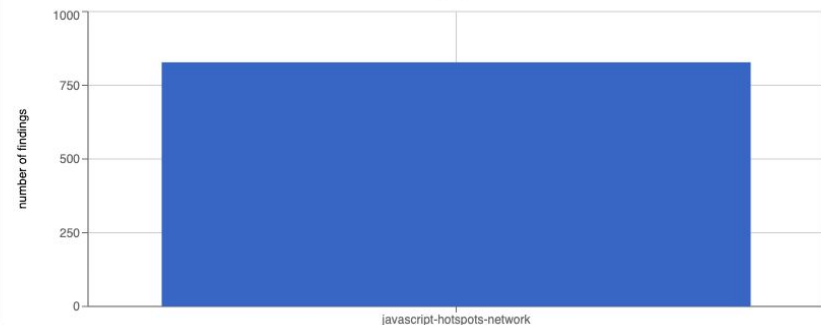
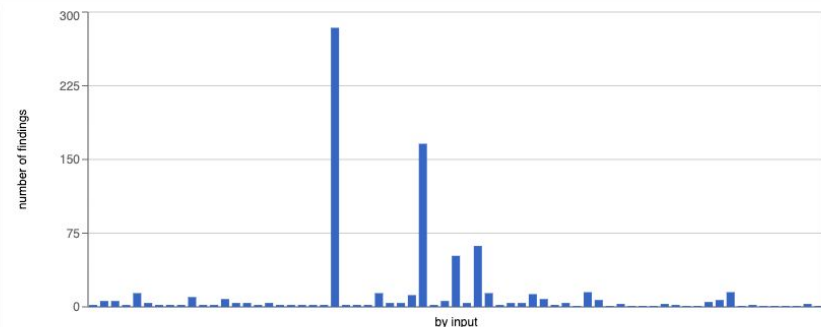
filter by commit hashes...

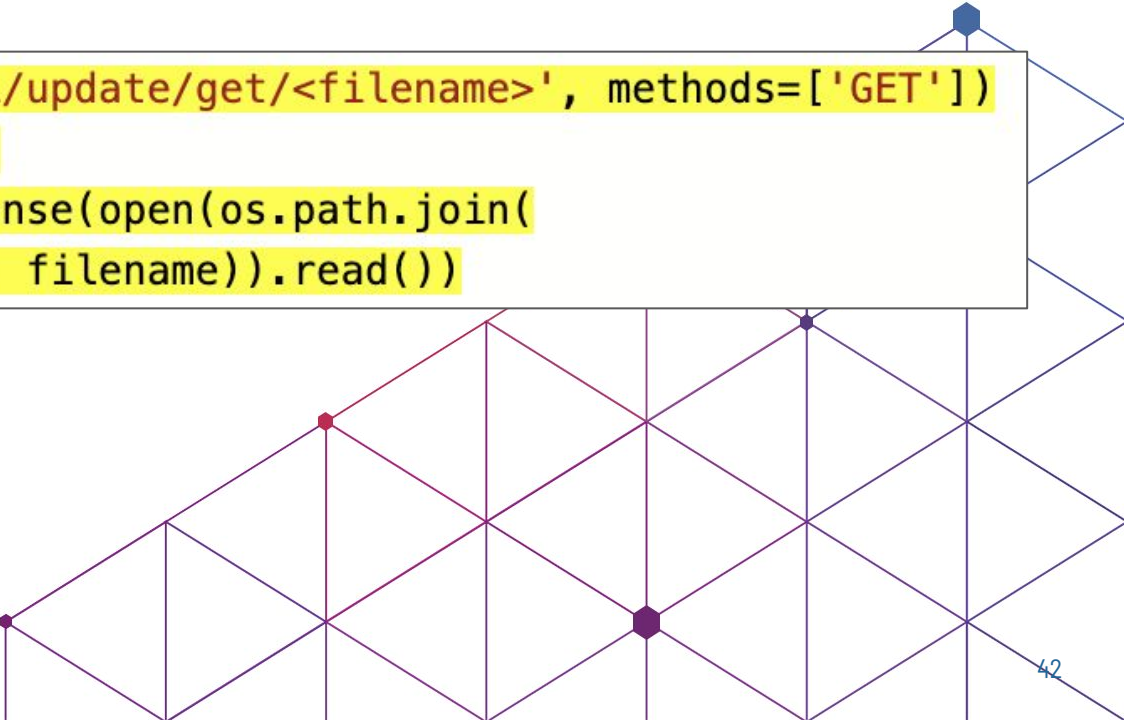
filter by check ids...

filter exclude path prefixes...

☐ Only Severity = ERROR

	Repository	Commit	Findings	Annotations	Action			
1	https://github.com/caolan/nodeunit	cd773a2	2		results			
2	https://github.com/trentm/node-bunyan	fe31b83	6		results			
3	https://github.com/kangax/html-minifier	51ce10f	6		results			
4	https://github.com/aheckmann/gm	e715cbd	2		results			
5	https://github.com/felixge/node-formidable	d23e560	14		results			
6	https://github.com/driverdan/node-XMLHttpRequest	97966e4	4		results			
7	https://github.com/defunctzombie/zuul	0a5644c	2		results			
8	https://github.com/intesso/connect-livereload	7c6ca1f	2		results			
9	https://github.com/chjj/blessed	eab243f	2		results			
10	https://github.com/request/request	212570b	10		results			
11	https://github.com/expressjs/express	e1b45eb	2		results			
12	https://github.com/moment/moment	13a61b2	2		results			
13	https://github.com/facebook/react	d862f0e	8		results			
14	https://github.com/gruntjs/grunt-contrib-watch	fc8458e	4		results			
15	https://github.com/karma-runner/karma	6235e68	4		results			
16	https://github.com/mishoo/UglifyJS2	70bb304	2		results			
17	https://github.com/webpack/webpack-dev-server	9d1c6d2	4		results			
18	https://github.com/jsdom/jsdom	699ed6b	2		results			
19	https://github.com/ember-cli/ember-cli	d6bbe89	2		results			
20	https://github.com/rwjlblue/ember-cli-inject-live-reload	5a37c1d	2		results			
21	https://github.com/NodeRedis/node_redis	a60261d	2		results			
22	https://github.com/visionmedia/superagent	67a5eee	2		results			
23	https://github.com/hock/hock	f6e319d	284		results			
24	https://github.com/senchalabs/connect	fa8916e	2		results			
25	https://github.com/BrowserSync/browser-sync	2191369	2		results			
26	https://github.com/facebook/jest	bc0f55d	2		results			
27	https://github.com/aws/aws-sdk-js	c9ac802	14		results			
28	https://github.com/websockets/ws	08c6c8b	4		results			
29	https://github.com/koajs/koa	817b498	4		results			
30	https://github.com/nodejs/readable-stream	4ba93f8	12		results			





```
50 @frontend.route('/api/update/get/<filename>', methods=['GET'])
51 def getZip(filename):
52     return make_response(open(os.path.join(
53         TEMPLATE_DIR, filename)).read())
```

Outline

1. Background
2. `grep` and Abstract Syntax Trees (ASTs)
3. Semgrep Examples!
4. Using Semgrep for Mass Scanning
- 5. Integration into CI/CD**
6. Semgrep Rules Registry

Integrations

- Enforce secure defaults + secure frameworks at CI time
 - Easy to add to CI as either a Docker container or Linux binary
 - JSON output → easy to integrate with other systems

Use in CI

`git pre-commit` [GitHub](#)  GitLab  CircleCI  AppVeyor  Travis

Add this snippet in your `.github/workflows/semgrep.yml`:

```
name: Semgrep
on: [push, pull_request]
jobs:
  semgrep:
    runs-on: ubuntu-latest
    name: Check
    steps:
      - uses: actions/checkout@master
```

Integrations

▼ Linters
on: pull_request

✓ super-linter

✓ pre-commit

✗ semgrep with managed policy

Linters / semgrep with managed policy
failed 1 hour ago in 1m 25s

▶ ✓ Set up job

▶ ✓ Pull returntocorp/semgrep-action:v1

▶ ✓ Run actions/checkout@v1

▼ ✗ Run returntocorp/semgrep-action@v1

```
GITHUB_EVENT_NAME -e GITHUB_SERVER_URL -e GITHUB_API_URL -e GITHUB_GRAPHQL_URL
-e ACTIONS_RUNTIME_URL -e ACTIONS_RUNTIME_TOKEN -e ACTIONS_CACHE_URL -e GITHUB_
"/home/runner/work/_temp/github_home":"/github/home" -v "/home/runner/work/_te
returntocorp/semgrep-action:v1
6 === detecting environment
7 | versions      - semgrep 0.17.0 on Python 3.8.5
8 | environment   - running in github-actions, triggering event is 'pull_request'
9 | semgrep.dev   - logged in as deployment #1
10 === setting up agent configuration
11 | using semgrep rules configured on the web UI
12 | using default path ignore rules of common test and dependency directories
13 | adding further path ignore rules configured on the web UI
14 | looking at 1 changed path
15 | found 1 file in the paths to be scanned
16 === looking for current issues in 1 file
17 | 1 current issue found
18 === looking for pre-existing issues in 1 file
19 | 1 pre-existing issue found
20 python.flask.security.injection.path-traversal-open.path-traversal-open
21                                     .py:459
22
23 459 | open(path).readlines(), mimetype="text/plain"
24
25     = Found request data in a call to 'open'. Ensure the request data is
26       validated or sanitized, otherwise it could result in path traversal
27       attacks.
28
29 === exiting with failing status
```

▶ ✓ Complete job

Outline

1. Background
2. `grep` and Abstract Syntax Trees (ASTs)
3. Semgrep Examples!
4. Using Semgrep for Mass Scanning
5. Integration into CI/CD
6. **Semgrep Rules Registry**

Community rule registry

community participation

- 100s of rules under development by r2c + community
- Rules contributed by [Damian Gryski](#), author of [Go Perfbook](#)
- **NodeJsScan v4 is powered by semgrep!**
- [Gosec](#) and [find-sec-bugs](#) checks have been ported - no compilation required 👍
- Rule ideas contributed by Django co-creator
- Independent security researchers via HackerOne & elsewhere



Community rule registry

semgrep.dev/registry ⇒ github.com/returntocorp/semgrep-rules

```
$ brew install semgrep  
$ semgrep --config=<url>
```


r2c

Go Java JavaScript Python

Default ruleset, by r2c

audit cookies correctness
crypto csrf injection security
spring xss xxe

r2c-ci

Go Java JavaScript Python

Scan for runtime errors, logic bus, and high-confidence security vulnerabilities....

CI cookies correctness crypto
csrf injection security spring
xss xxe logic logic bugs

r2c-security-audit

Ruby JavaScript Go Java C

Scan code for potential security issues that require additional review. Recommended for tea...

security audit xxe injection
deserialization xss jwt csrf
crypto

Languages and Frameworks

Get security coverage for the languages and frameworks you use.

minusworld.ruby-all

python

Python

Default ruleset for Python, by r2c

security correctness

javascript

JavaScript

Default ruleset for JavaScript, by r2c

security correctness

```
$ semgrep --config=https://semgrep.dev/p/r2c
```

Coming Soon

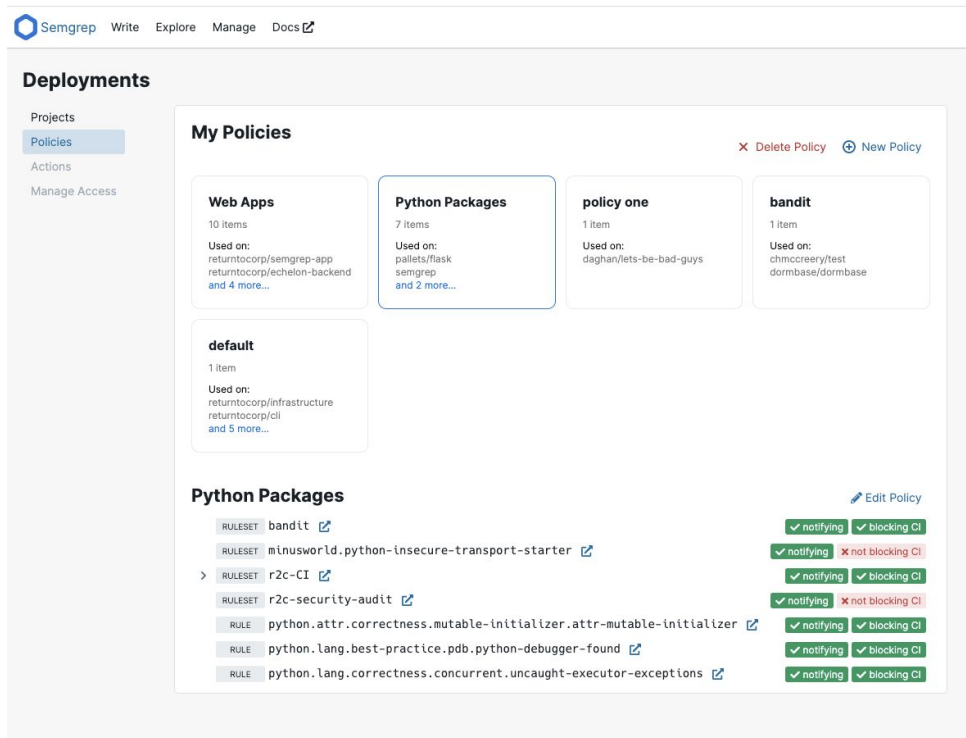
More rules! + prevention cheatsheets

Semgrep Community!

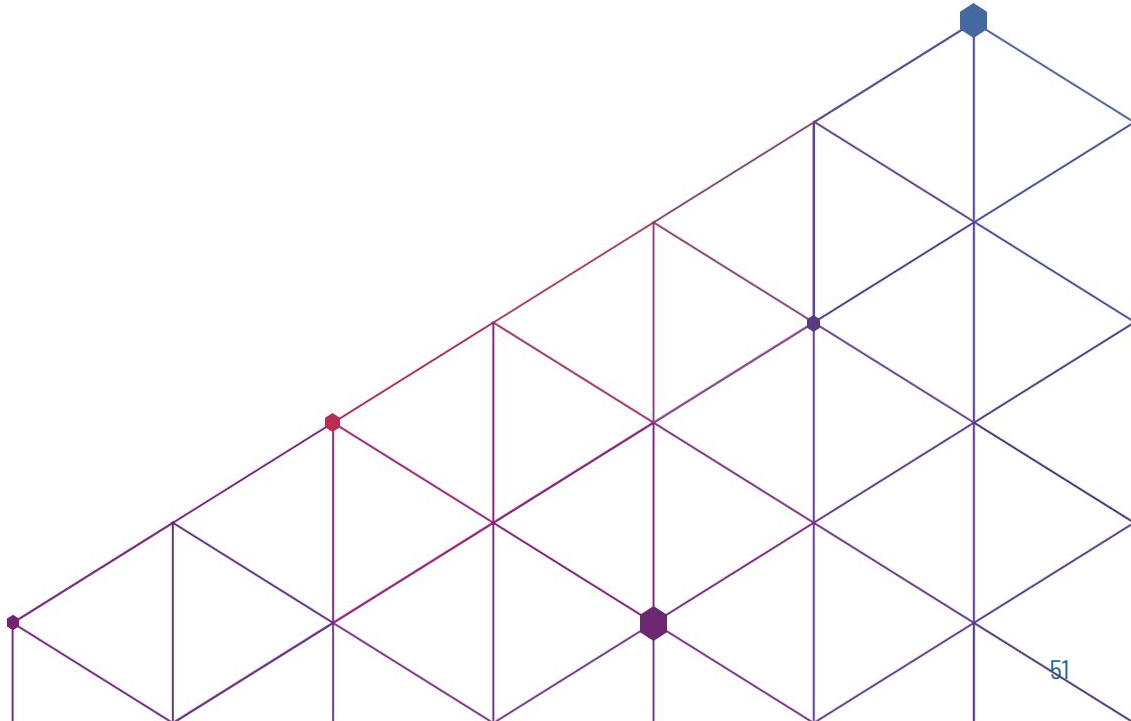
Centrally manage Semgrep on your repos!

VS Code extension (in beta)!

Tainting + constant propagation!



Wrapping Up





Semgrep

lightweight static analysis for many languages

Locally:

1. `(brew or pip) install semgrep`
2. `semgrep --config=r2c .`

Online editor:

- semgrep.live

Colleen Dai | colleen@returntocorp.com

r2c.dev |  [@r2cdev](https://twitter.com/r2cdev)

<https://r2c.dev/survey> ← plz :)

