



# Detect complex code patterns using semantic grep

Drew Dennison | r2c

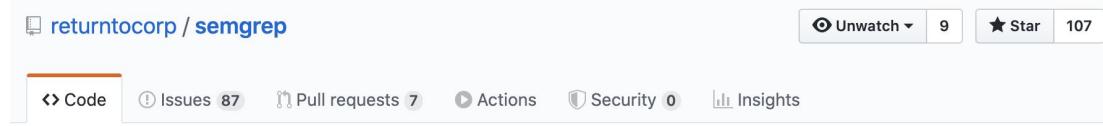
 [@drewdennison](https://twitter.com/drewdennison)



Slides are posted at [bit.ly/semgrep-toronto](https://bit.ly/semgrep-toronto)

# Tl;dw – This Talk

- Writing secure code is hard
- Security is shifting towards code
- Need something, fast, code-aware, flexible, powerful... **open source!**
- Enter: [semgrep](#)



## Use to:

like grep but for code: fast and syntax-aware semantic code pattern for many languages <https://semgrep.live>

- Enforce org / code specific patterns and best practices
- Find security bugs
- Scan every PR for vulnerabilities

# whois?

## Presenting:

Drew Dennison, co-founder @ r2c

BS in CompSci from MIT, ex-Palantir dev

r2c

We're a static analysis startup based in SF and we really care about software security.





Chicago



San Francisco

# Talk Outline

1. Writing secure code is hard
2. Getting started with semgrep
3. Tutorial
4. Using it yourself

# Top 10 from Open Web Application Security Project

1. Injection
2. Broken authentication
3. Sensitive data exposure
4. XML XXE
5. Broken access control
6. Security misconfiguration
7. Cross-site scripting (XSS)
8. Insecure deserialization
9. Components with known vulnerabilities
10. Insufficient logging & monitoring



# But wait, there's more!

## OWASP Top 10 for webapps

1. Injection
2. Broken authentication
3. Sensitive data exposure
4. XML XXE
5. Broken access control
6. Security misconfiguration
7. Cross-site scripting (XSS)
8. Insecure deserialization
9. Components with known vulnerabilities
10. Insufficient logging & monitoring

## OWASP Top 10 for APIs

1. BOLA (Broken Object Level Authorization)
2. Broken Authentication
3. Excessive Data Exposure
4. Lack of Resources & Rate Limiting
5. BFLA (Broken Function Level Authorization)
6. Mass Assignment
7. Security Misconfiguration
8. Injection
9. Improper Assets Management
10. Insufficient Logging & Monitoring



# #6: Security Misconfiguration



**ars** TECHNICA

BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE STUFF

SPEAK NO EVIL —

## Here's the Netflix account compromise Bugcrowd doesn't want you to know about [Updated]

Weakness allows attackers to steal browser cookies used to authenticate Netflix users.

DAN GOODIN - MAR 23, 2020 10:37 PM UTC

<https://arstechnica.com/information-technology/2020/03/bugcrowd-tries-to-muzzle-hacker-who-found-netflix-account-compromise-weakness/>



# It's really hard to write secure code

```
set_cookie(key, value='', max_age=None, expires=None,  
            httponly=False, samesite=None)
```

Sets a cookie. The parameters are the same as in the `Cookie` class, but it accepts unicode data, too.

A warning is raised if the size of the cookie header exceeds 4096 bytes. The `max_age` parameter can be negative, in which case the cookie will be deleted.

- Parameters:**
- **key** – the key (name) of the cookie to be set.
  - **value** – the value of the cookie.
  - **max\_age** – should be a number of seconds, or `None` (default) if the cookie should last only as long as the client's browser session.
  - **expires** – should be a `datetime` object or UNIX timestamp.
  - **path** – limits the cookie to a given path, per default it will span the whole domain.
  - **domain** – if you want to set a cross-domain cookie. For example,



GOTCHA

# Cookies



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

How might we detect this? ⇒ REGEX

```
grep -RE 'response\.set_cookie\(' path/to/code
```

# Code Equivalence

```
response.set_cookie("username", "DrewDennison")
```

Default

```
from flask.response import set_cookie as sc  
sc("username", "DrewDennison")
```

Import and rename

```
response.set_cookie("username", "Drew", secure=True, path="/")
```

Keyword arguments

```
response.set_cookie(  
    "username", "DrewDennison",  
    secure=True, samesite=Secure, httponly=True)
```

Multiline

```
# response.set_cookie("username", "DrewDennison", secure=True)
```

Commented out code

# Regex Matches

```
response.set_cookie("username", "DrewDennison")
```

```
from flask.response import set_cookie as sc  
sc("username", "DrewDennison")
```

1 True Positive

```
response.set_cookie("username", "Drew", secure=True, path="/")
```

```
response.set_cookie(  
    "username", "DrewDennison",  
    secure=True, samesite=Secure, httponly=True)
```

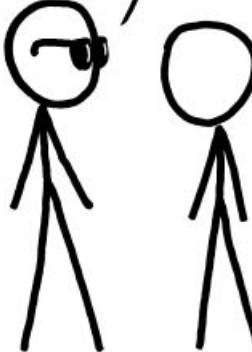
1 False Negative

```
# response.set_cookie("username", "DrewDennison", secure=True)
```

IF YOU'RE HAVIN' PERL  
PROBLEMS I FEEL  
BAD FOR YOU, SON-



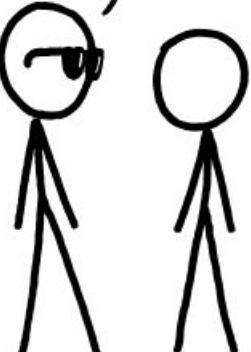
I GOT 99  
PROBLEMS,



SO I USED  
REGULAR  
EXPRESSIONS.



NOW I HAVE  
100 PROBLEMS.



# Code is not a string, it's a tree



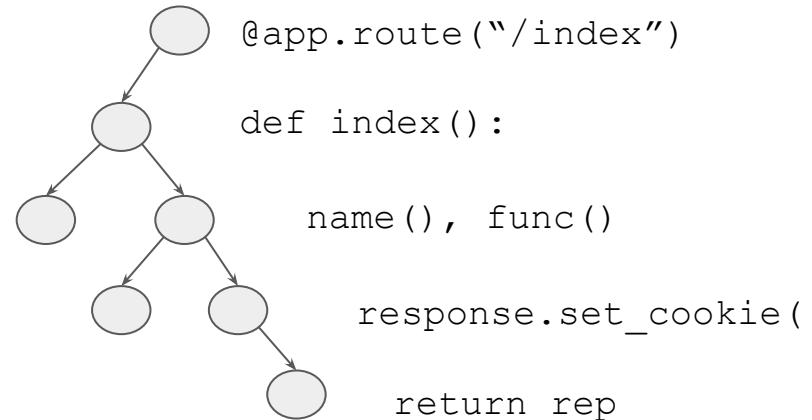
**string**

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

**!=**



**tree**



# Linters & SAST: Solution?

- NodeJSScan, dlint, Bandit, Flake8, eslint, tslint, pylint, Golint, rubocop, among others...
- Have to become an **expert in every AST syntax** for every language my team uses
- **Commercial SAST tools?**
  - Expensive
  - Hard to tune

```
import ast
from typing import List

from flake8_boto3 import __version__
from flake8_boto3.dumb_scope_visitor import DumbScopeVisitor

BOTOS_NAME = "boto3"
BAD_KEYWORDS = ("aws_access_key_id", "aws_secret_access_key", "aws_session_token")

class AuthTokenVisitor(DumbScopeVisitor):
    name = "2c-boto3-hardcoded-access-token"

    def __init__(self):
        self.report_nodes: List[ast.Call] = []
        self.using_boto3 = False
        super(AuthTokenVisitor, self).__init__()

    def _message(self, node):
        return f"({self.name}) Hardcoded access token detected. Consider using a config file or environment variables."

    def _make_report(self, node):
        return ("node": node, "message": self._message(node))

    def _validate_token(self, keyword_arg: str, str_nodes: List[ast.Str]) -> bool:
        ...
        Return true if any value:
        1. is not a repetition of the same character, e.g., '---' or 'XXXX'
        2. does not contain the word "access" or "secret", e.g., "aws_access_token"
        3. does not have a space, e.g., "your token here"
        4. is 10 or more chars
        ...
        str_values: List[str] = [self._get_symbol_value(sn) for sn in str_nodes]
        return any([
            len(set(val)) != 1
            and "secret" not in val
        ])

    def visit_Call(self, call: ast.Call):
        keywords = call.keywords
        for keyword in keywords:
            if keyword.arg in BAD_KEYWORDS:
                if isinstance(keyword.value, ast.Str) and self._validate_
                    keyword.arg,
                    [keyword.value]
                ):
                    self.report_nodes.append(self._make_report(call))
                    break # only report this call once even if (likely)
        names = {import_name
        for fqn in names:
            if fqn.name == BOTOS_NAME:
                self.using_boto3 = True

    def visit_ImportFrom(self, import_from: ast.ImportFrom):
        if import_from.module == BOTOS_NAME:
            self.using_boto3 = True
```

```
flask.response.set_cookie(<whatever>)
```

# Complexity



# **semgrep** - search code the way you write it

```
$ brew install returntocorp/semgrep/semgrep  
$ semgrep -e 'flask.response.set_cookie(...)' -l python /path/to/my/project
```

⇒ Free & open source



First version of semgrep (sgrep) written at Facebook, used to enforce almost one thousand rules

Yoann Padoleau, original sgrep author and first program analysis hire at FB, joined r2c last year. Previously PhD @ INRIA,  
[coccinelle.lip6.fr](http://coccinelle.lip6.fr)

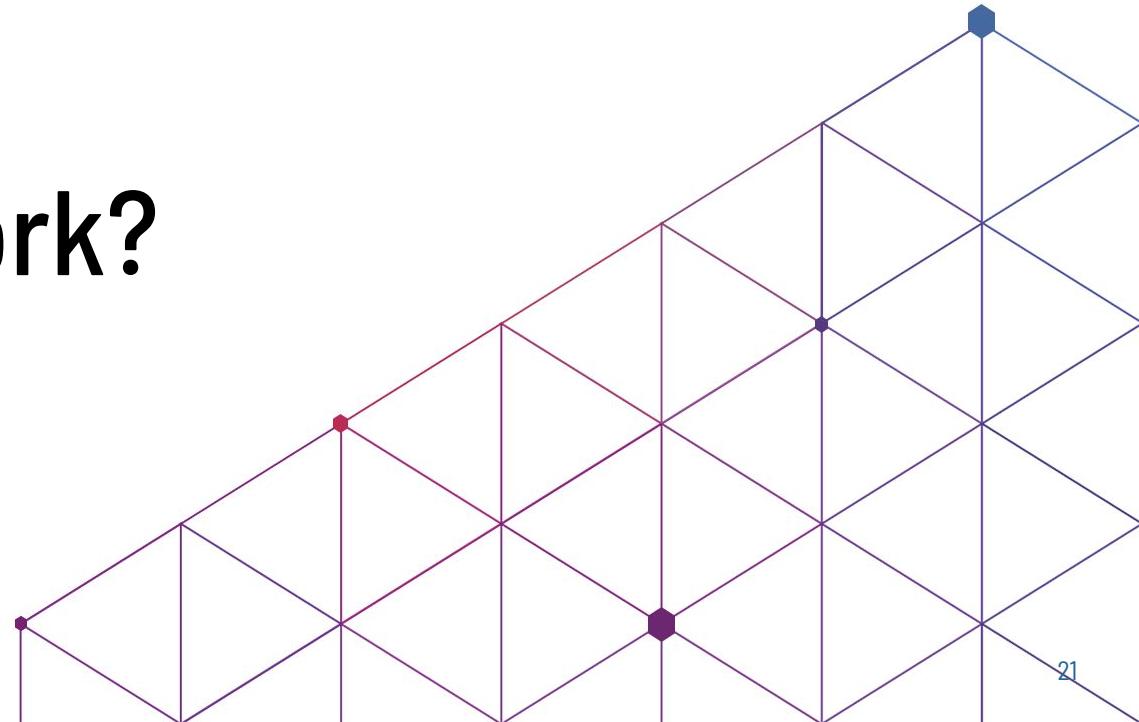
# Getting Started

[semgrep.dev](https://semgrep.dev) ⇒ [github.com/returntocorp/semgrep](https://github.com/returntocorp/semgrep)

The screenshot shows the GitHub repository page for `returntocorp / semgrep`. The page includes the following details:

- Code** tab is selected.
- Issues**: 87
- Pull requests**: 7
- Actions**
- Security**: 0
- Insights**
- Settings**
- Unwatch** (9 notifications)
- Unstar** (107 stars)
- Fork** (14 forks)
- Description: "like grep but for code: fast and syntax-aware semantic code pattern for many languages" with a link to <https://semgrep.live>.
- Topics**: static-analysis, ocamli, grep-like, semgrep, metavariables, Manage topics
- Statistics**: 561 commits, 17 branches, 0 packages, 53 releases, 14 contributors, LGPL-2.1 license.
- Branches**: develop (selected), New pull request.
- Actions**: Create new file, Upload files, Find file, Clone or download.
- Pull Requests**: brendongo Merge pull request #569 from returntocorp/release-0.5.0b2 ... (status: Latest commit bf3f33b 2 days ago).
- Issues**: .bento, Rename sgrep to semgrep in other markdown files (#542) (status: 3 days ago).

# How does it work?



# Tutorial: Golang servers not serving website over TLS

```
func main() {
    http.HandleFunc("/index", Handler)
    http.ListenAndServe(":80", nil)
}
```

⇒ <https://semgrep.live/dyP>

Full Solution: <https://semgrep.live/Zy7>

# Tutorial: Node Exec

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

⇒ <https://semgrep.live/Xnw>

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

# Tutorial: Injection Sending File

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

⇒ <https://semgrep.live/7dv>

Full Solution: <https://semgrep.live/EN8>

# Tutorial: problem

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

⇒ <https://semgrep.live/8dJ>

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

## Tutorial: exit()

```
if username == "DrewDennison":  
    print("n00b")  
    exit(1)  
else:  
    print("l33t")
```

⇒ <https://semgrep.live/z3P>

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

# Tutorial: Custom Logic for your code

```
/*
 * 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/LNX>

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

# Tutorial: 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>

**YOU REALLY EXPECT  
ME TO WRITE MY OWN RULES?**

**IT'S EXHAUSTING  
STAYING HOME ALL DAY**

# part 2: registry <sup>beta</sup>

# community rule registry

⇒ [github.com/returntocorp/semgrep-rules](https://github.com/returntocorp/semgrep-rules)

```
$ brew install returntocorp/semgrep/semgrep
$ semgrep --config=<folder of yaml files>
```

## community participation

- 100s of rules under development by r2c + community
- **NodeJsScan author ported 95% their rules**
- Rule ideas contributed by Django co-creator
- Suggestions by Flask team
- Independent security researchers

# community rule registry

## registry

secur

Scan Code

r2c.python.djangoproject.security  
r2c.python.djangoproject.security.audit  
r2c.python.djangoproject.security.audit.csrf-exempt  
r2c.python.djangoproject.security.audit.query-set-extra  
r2c.python.djangoproject.security.audit.raw-query  
r2c.python.djangoproject.security.audit.secure-cookies  
r2c.python.djangoproject.security.globals-misuse-code-execution  
r2c.python.djangoproject.security.injection  
r2c.python.djangoproject.security.injection.command-injection-os-system  
r2c.python.djangoproject.security.injection.mass-assignment  
r2c.python.djangoproject.security.injection.open-redirect  
r2c.python.djangoproject.security.injection.path-traversal-file-name  
r2c.python.djangoproject.security.injection.path-traversal-join  
r2c.python.djangoproject.security.injection.path-traversal-open  
r2c.python.djangoproject.security.injection.reflected-data-httpproxy  
r2c.python.djangoproject.security.injection.reflected-data-httpproxybadrequest  
r2c.python.djangoproject.security.injection.request-data-fileresponse  
r2c.python.djangoproject.security.injection.request-data-write  
r2c.python.djangoproject.security.injection.sql  
r2c.python.djangoproject.security.injection.sql.sql-injection-extra  
r2c.python.djangoproject.security.injection.sql.sql-injection-rawsql  
r2c.python.djangoproject.security.injection.sql.sql-injection-using-raw  
r2c.python.djangoproject.security.injection.ssr-injection-requests  
r2c.python.djangoproject.security.injection.ssr-injection-urllib  
r2c.python.djangoproject.security.injection.user-eval

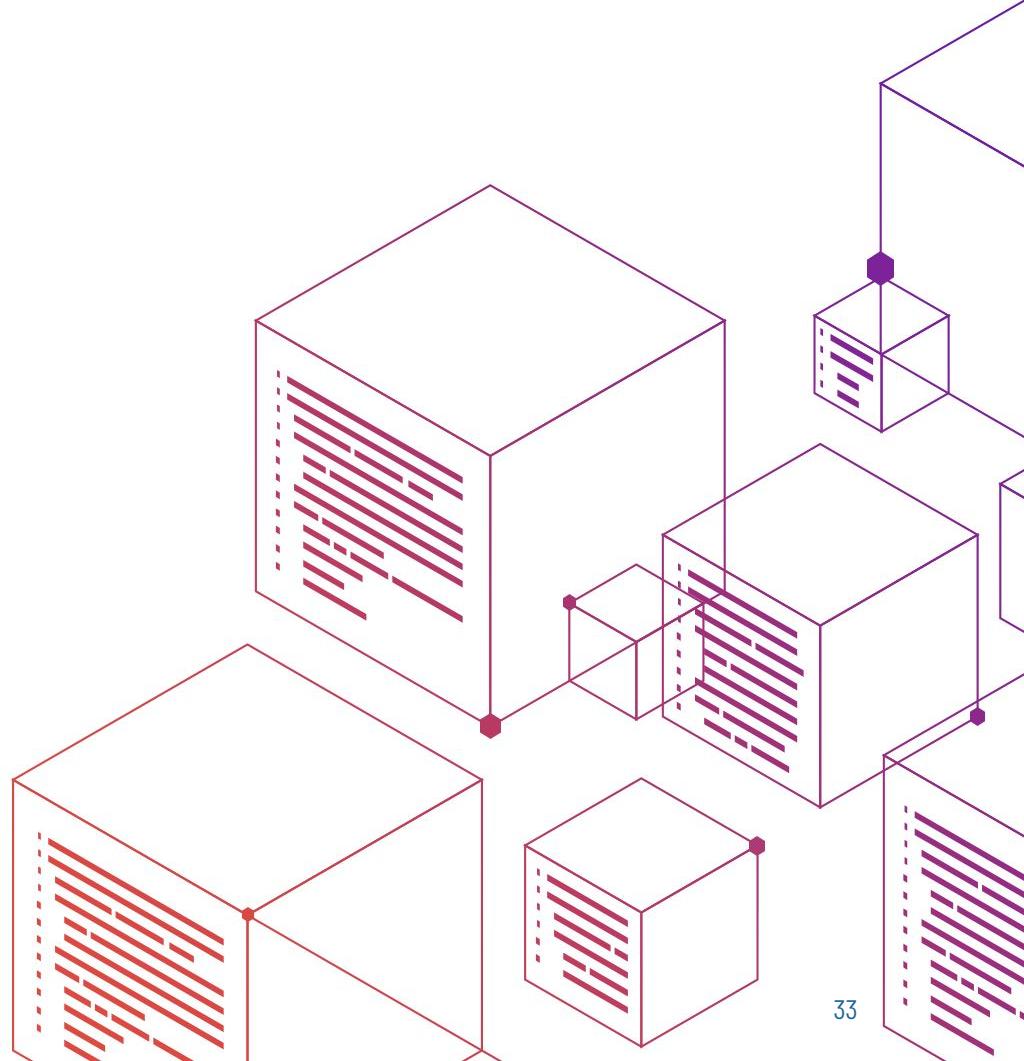
## registry > r2c.javascript.jwt-none-alg.jwt-none-alg

There is 1 rule in this rule pack

```
rules:  
- id: r2c.javascript.jwt-none-alg.jwt-none-alg.jwt-none-alg  
  message: None algorithm allowed for JWT token  
  languages: [javascript]  
  severity: ERROR  
  patterns:  
  - pattern-either:  
    # jsonwebtoken  
  - pattern: |  
    var $JWT = require("jsonwebtoken");  
    ...  
    var $T = $JWT.verify($P, $X, {algorithms:[..., 'none', ...]}, ...);  
  - pattern: |  
    var $JWT = require("jsonwebtoken");  
    ...  
    $T = $JWT.verify($P, $X, {algorithms:[..., 'none', ...]}, ...);  
  - pattern: |  
    var $JWT = require("jsonwebtoken");  
    ...  
    $JWT.verify($P, $X, {algorithms:[..., 'none', ...]}, ...);  
  - pattern: |  
    var $JOSE = require("jose");
```



# Tuning Rules



# 5000x Parallel Map Reduce

dev/semgrep

l9 version: 0.5.1

input set: awesome-go-2020-02-20/0.0.1

Re-run job

Download job

[job status](#) [results](#) [parameters](#)

1319/1319 repos  
repos analyzed

0.91%  
error rate

5/5/2020, 3:46:29 PM  
submission timestamp

[View results](#)

100% completed | done

errors

[DependentAnalyzerFailed \(1\)](#) [Error \(11\)](#)



# reduce false positives at scale

Rule tuned with map-reduce platform over thousands of real git repos

```
rules:
  - id: eqeq-is-bad
    patterns:
      - pattern-not-inside: |
          def __eq__(...):
            ...
      - pattern-not-inside: assert(...)
      - pattern-not-inside: assertTrue(...)
      - pattern-not-inside: assertFalse(...)
      - pattern-either:
          - pattern: $x == $x
          - pattern: $x != $x
      - pattern-not: 1 == 1
    message: "useless comparison operation `\$x == \$x` or `\$x != \$x`;
if testing for floating point NaN, use `math.isnan`, or `cmath.isnan`
if the number is complex."
    languages: [python]
    severity: ERROR
```



Real bugs

[1]

<https://github.com/secdev/scapy/blob/8066e9d87165b7c5387cc13e141f58cc2603dc1a/scapy/contrib/isotp.py#L364>

[2]

<https://github.com/apache/libcloud/blob/6540c127199f92465c4917a974f1fb06bc71d27/libcloud/compute/drivers/cloudsigma.py#L612>

[3]

[https://github.com/Azure/azure-sdk-for-python/blob/76835cddac9e6996dace0ee686ae5a7c446c5f8/sdk/cosmos/azure-cosmos/azure/cosmos/routing/routing\\_map\\_provider.py#L97](https://github.com/Azure/azure-sdk-for-python/blob/76835cddac9e6996dace0ee686ae5a7c446c5f8/sdk/cosmos/azure-cosmos/azure/cosmos/routing/routing_map_provider.py#L97)

Let me know if would like to help beta-test our  
verified rules

<https://r2c.dev/survey>

# Integrations

- Find bugs in code. Bonus points for CVEs!
- Enforce secure defaults + secure frameworks at CI time
  - Easy to add to CI as either docker container or linux binary
  - JSON output

## CircleCI Config

```
jobs:  
  build:  
    docker:  
      - image: returntocorp/semgrep:develop  
    working_directory: /home/repo  
    steps:  
      - checkout  
      - run: semgrep --config=https://sgrep.live/c/r/r2c.python.django
```

## Github Action

### semgrep action

This action runs [semgrep](#) and returns the output

#### Inputs

##### config

The config file|directory|yaml\_url|tar|url|registry\_name .

##### output

The output arg file|url

##### targets

The target(s) to scan

##### error

If true will exit 1 which will break the build.

# Security + Developers = ❤



Code Issues Pull requests Actions Security Insights Settings

## Security overview

Overview

Security policy

Security advisories 0

Dependency alerts 0

● Security View details Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes

410 <<<<< HEAD (Current Change)

411 => this.updateSizeClasses();

412 => this.multiCursorModifier();

413 => this.contentDisposables.push(this.configurationService.onDidU

414 =====

415 => this.toggleSizeClasses();

416 >>>> Test (Incoming Change)

417 => if (input.onReady) {

418 => => input.onReady(innerContent);

419 => => }

420 => => this.scrollbar.scanDOMNode();

421 => => this.loadTextEditorViewState(input.getResource());

422 => => this.updatedScrollPosition();

423 => => };

424 => }



# Semgrep Core Features Coming Soon

More code equivalences!

$$\$X + \$Y \iff \$Y + \$X$$

Integrate typing information

```
new Buffer($X:int, ...)
```

```
($O:XmlHttpRequest).get(...)
```

A screenshot of a GitHub repository page for `returntocorp / semgrep`. The page shows various metrics: 95 issues, 7 pull requests, 11 actions, 0 security vulnerabilities, 229 insights, and 19 forks. Below these are tabs for Labels and Milestones, with 'Milestones' being active. A green button labeled 'New milestone' is visible. Under the milestones section, there is one open milestone titled '0.7.x'. This milestone has a progress bar indicating it is 37% complete, with 5 open issues and 3 closed issues. It was due by May 19, 2020, and was last updated less than a minute ago. There are buttons for Edit, Close, and Delete.

# Thanks



We have a bunch of rules for specific frameworks for OWASP Top 10 issues

⇒ <https://github.com/returntocorp/semgrep-rules>

Or take our survey about the talk and we'll email them to you

<https://r2c.dev/survey>

# May the Sith be with you!





# Semgrep

**open-source** code-analysis that feels like grep

## Semgrep:

1. Easily match interesting code patterns
2. Fast! 100K LOC in seconds
3. Python, JavaScript, Golang, Java

## Run it now!

1. [semgrep.live](https://semgrep.live)
2. brew install  
returntocorp/semgrep/semgrep
3. docker run --rm -v "\${PWD}:/home/repo"  
returntocorp/semgrep

[Drew Dennison](https://drewdennison.com) | [@drewdennison](https://twitter.com/drewdennison) | [drew@r2c.dev](mailto:drew@r2c.dev)

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

slides are posted at [bit.ly/semgrep-toronto](https://bit.ly/semgrep-toronto)

