Detect complex code patterns using semantic grep

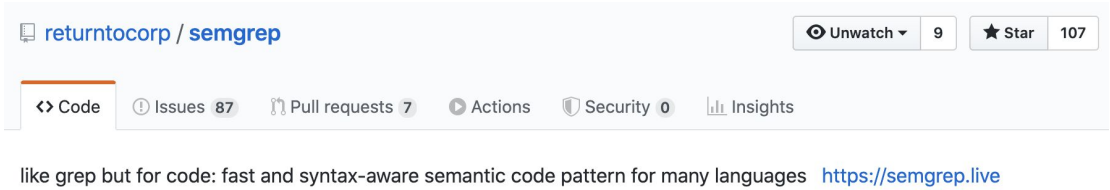
Drew Dennison | r2c

@drewdennison

Slides are posted at 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:

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

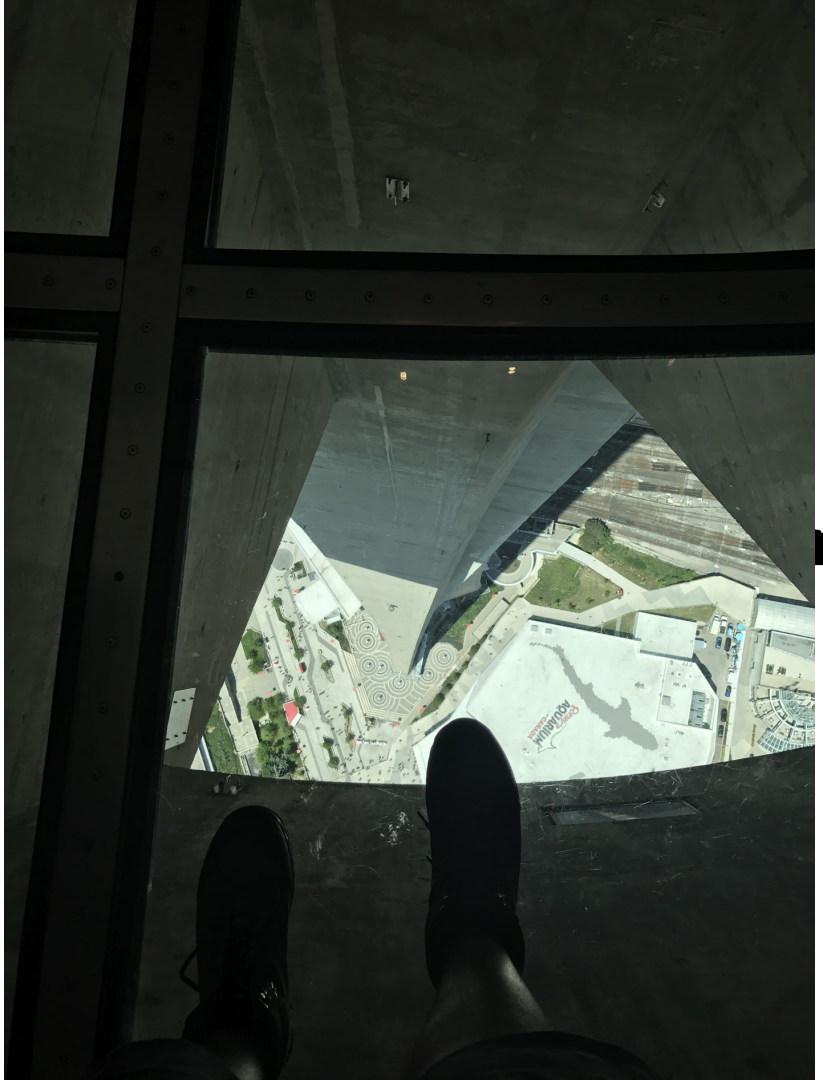
whois?

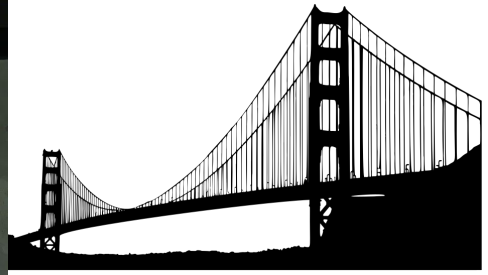
Presenting:
Drew Dennison, co-founder @ r2c

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

https://owasp.org/www-project-top-ten/

But wait, there's more!

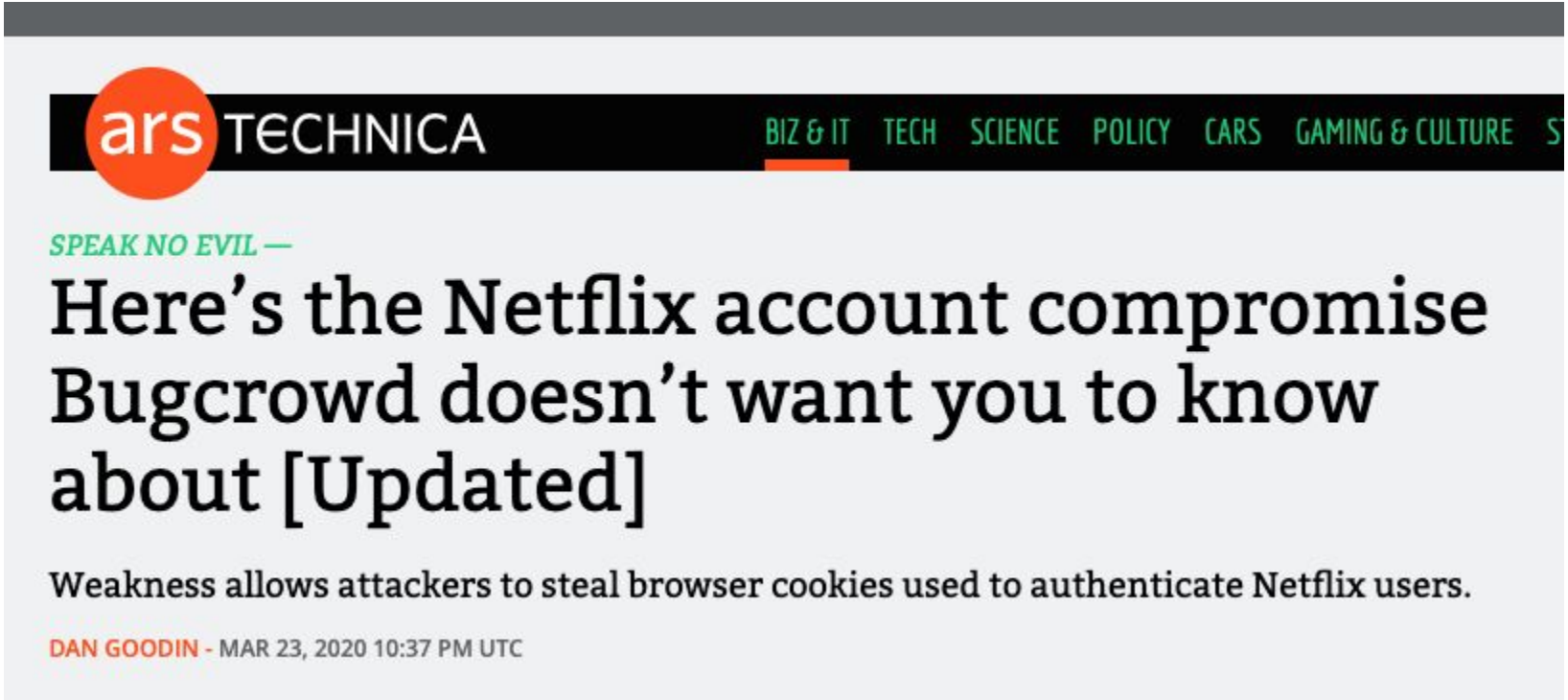
OWASP Top 10 for *webapps*

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 S

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=N*
httponly=False, *samesite=None*)

Sets a cookie. The parameters are the same as in the co rary but it accepts unicode data, too.

A warning is raised if the size of the cookie header exce ll be set.

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

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")

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

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

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

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

Default

Import and rename

Keyword arguments

Multiline

Commented out code

Regex Matches

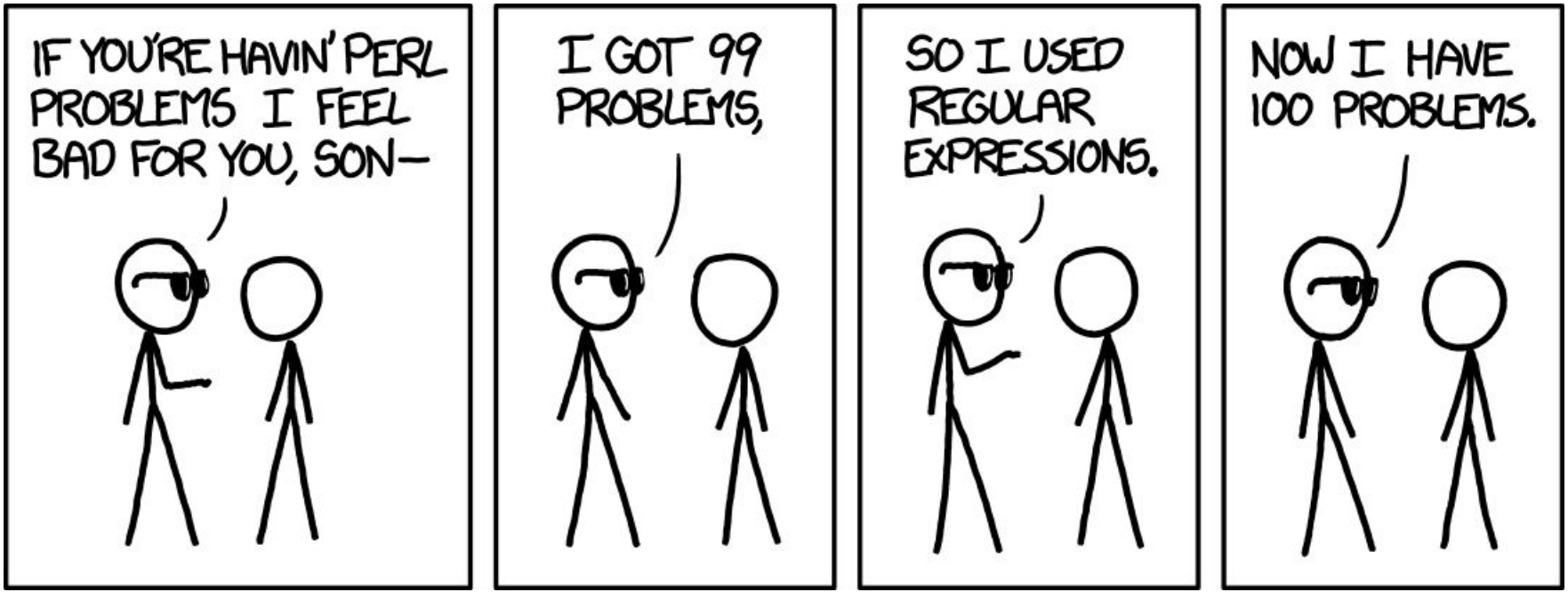
```
response.set_cookie("username","DrewDennison")

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

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

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

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

1 True Positive

3 False Positives

1 False Negative

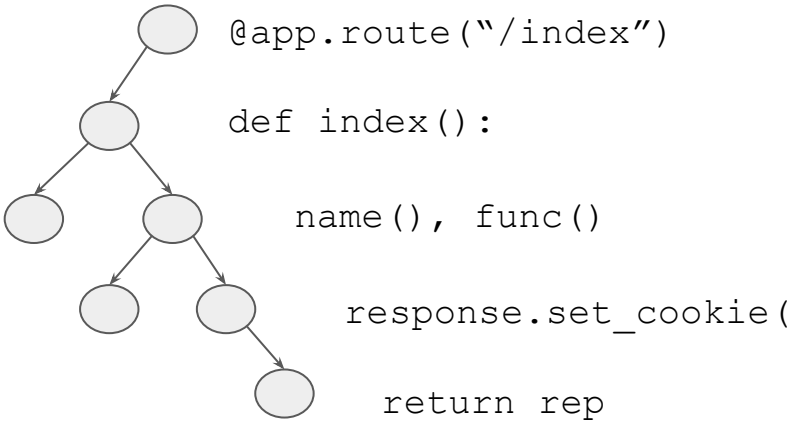
xkcd 1171


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 **!=** **tree**

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



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

```python
import ast
from typing import List

from flake8_boto3 import __version__
from flake8_boto3.dumb_scope_visitor import DumbScopeVisitor

BOTO3_NAME = "boto3"
BAD_KEYWORDS = {"aws_access_key_id", "aws_secret_access_key", "aws_session_token"}


class AuthTokenVisitor(DumbScopeVisitor):
    name = "r2c-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 16 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
```

```python
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)
```

```python
        names = _import.names
        for fqn in names:
            if fqn.name == BOTO3_NAME:
                self.using_boto3 = True

    def visit_ImportFrom(self, import_from: ast.ImportFrom):
        if import_from.module == BOTO3_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
```

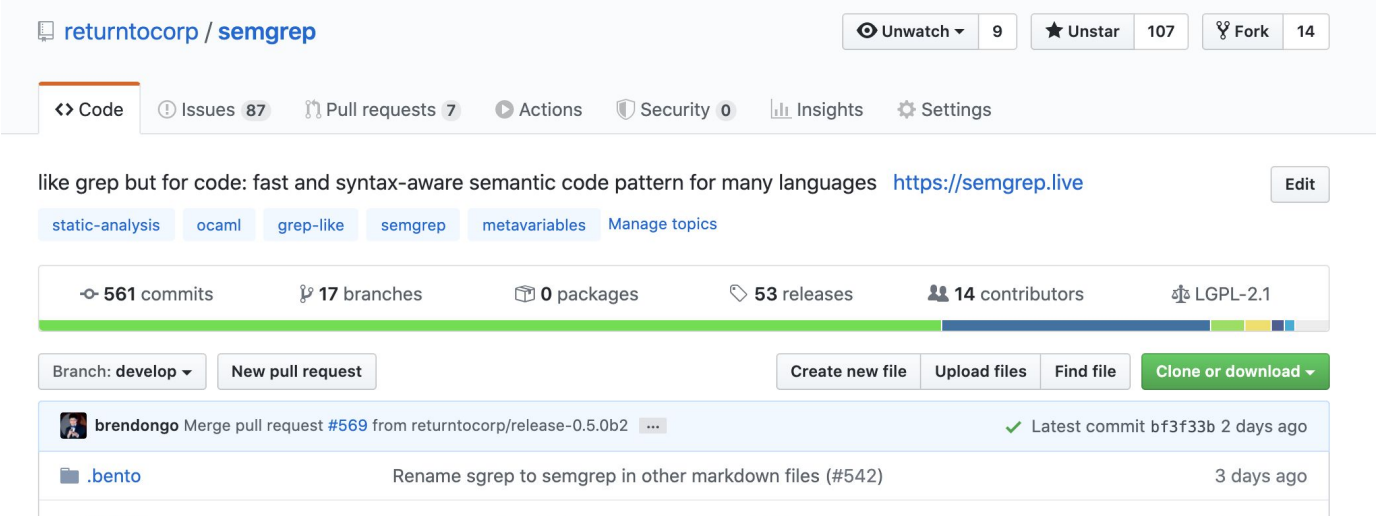
⇒ Free & open source

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

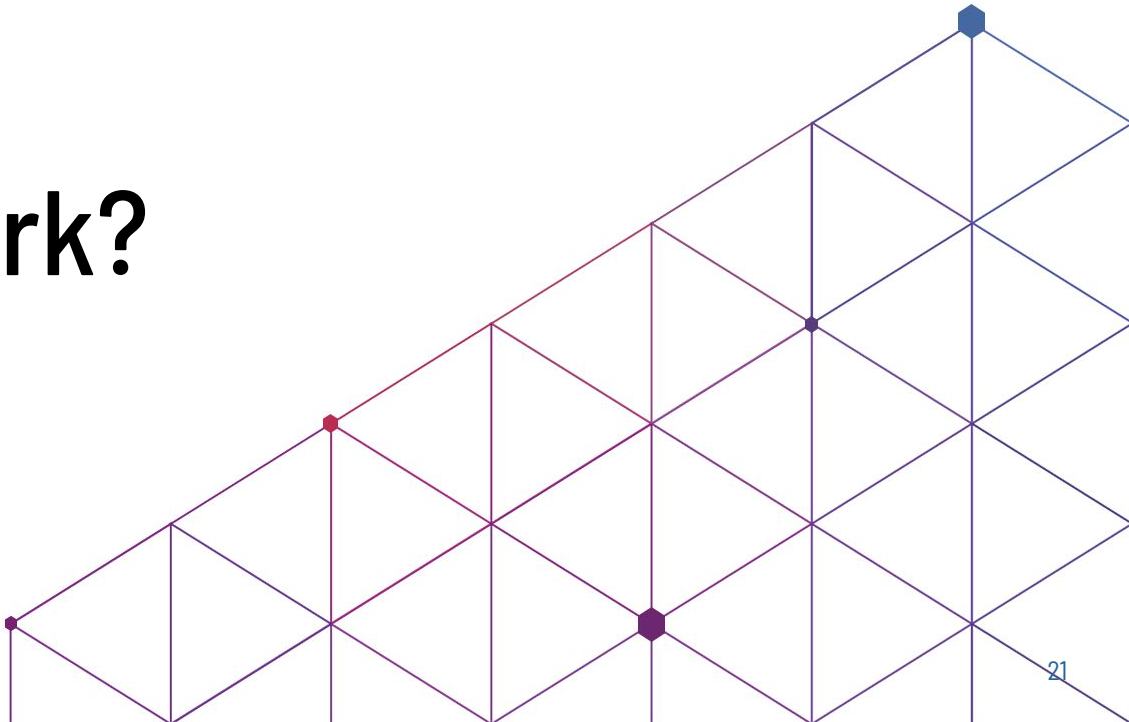
Yoann Padioleau, original sgrep author and first program analysis hire at FB, joined r2c last year. Previously PhD @ INRIA, coccinelle.lip6.fr

Getting Started

semgrep.dev ⇒ github.com/returntocorp/semgrep



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("133t")
```

⇒ 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

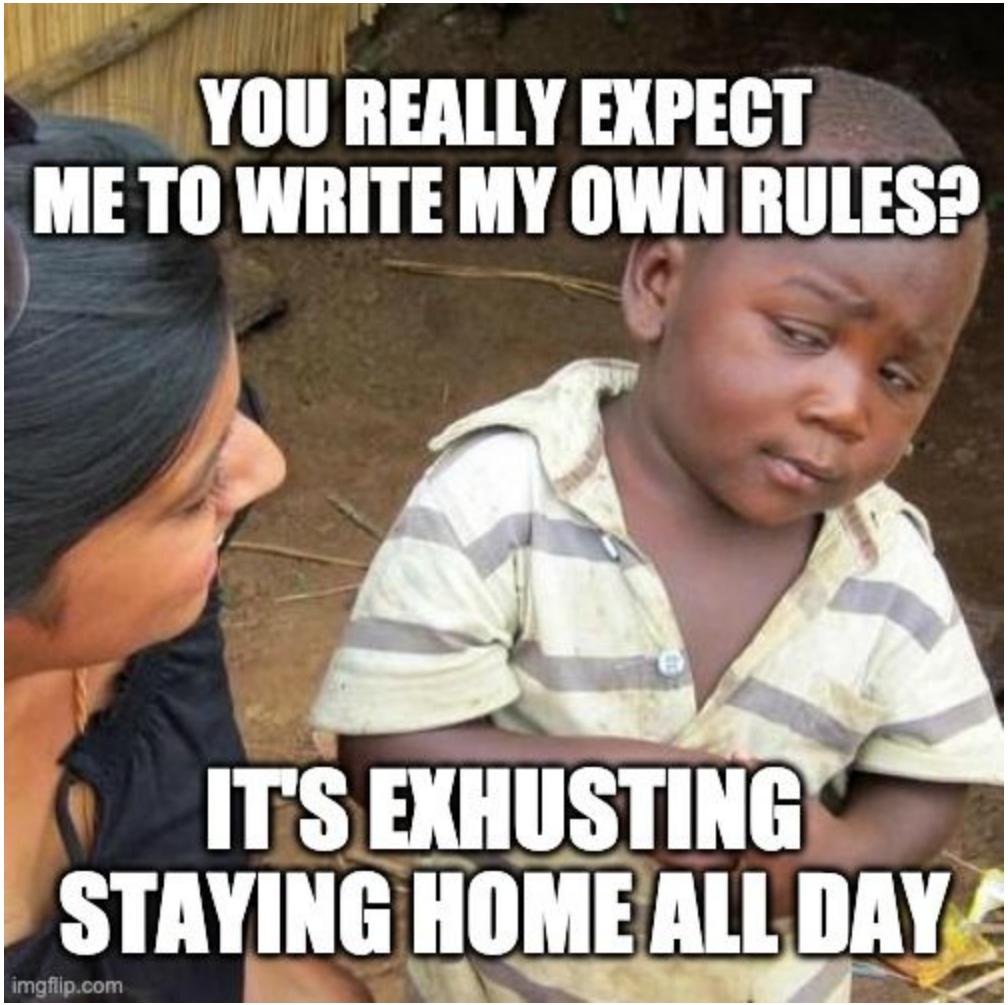
Full Solution: https://semgrep.live/kle

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 EXHUSTING
STAYING HOME ALL DAY
imgflip.com

part 2: registry beta

community rule registry

⇒ 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

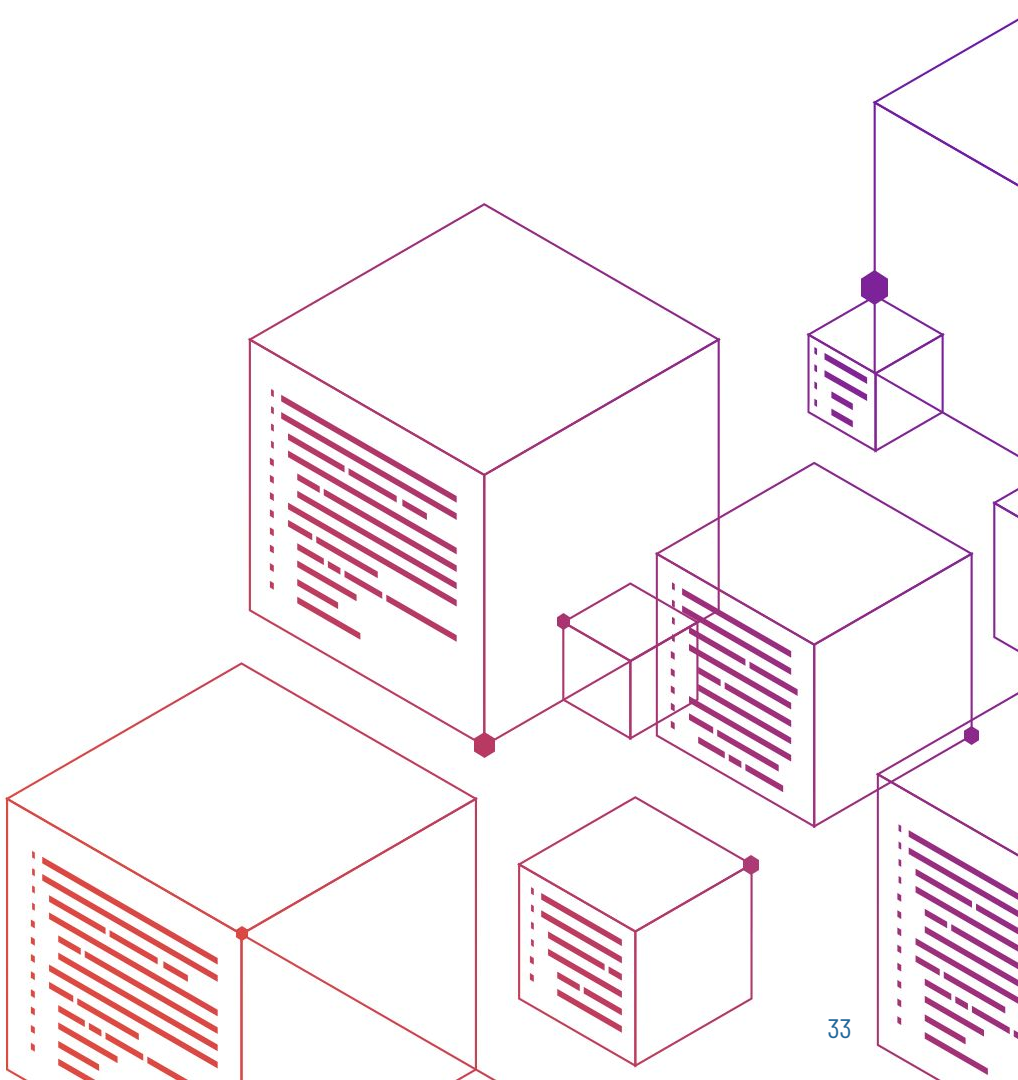
r2c.python.django.security
r2c.python.django.security.audit
r2c.python.django.security.audit.csrf-exempt
r2c.python.django.security.audit.query-set-extra
r2c.python.django.security.audit.raw-query
r2c.python.django.security.audit.secure-cookies
r2c.python.django.security.globals-misuse-code-execution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

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

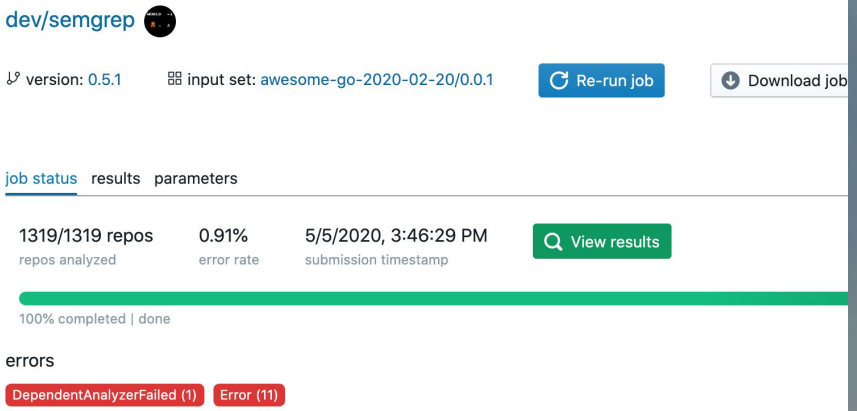
There is 1 rule in this rule pack

```
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



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/6540c127199f92465c4917a974f1fbd06bc71d27/libcloud/compute/drivers/cloudsigma.py#L612

[3] https://github.com/Azure/azure-sdk-for-python/blob/76835cddacb9e6996dace0ee686ae5a7c446c5f8/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

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

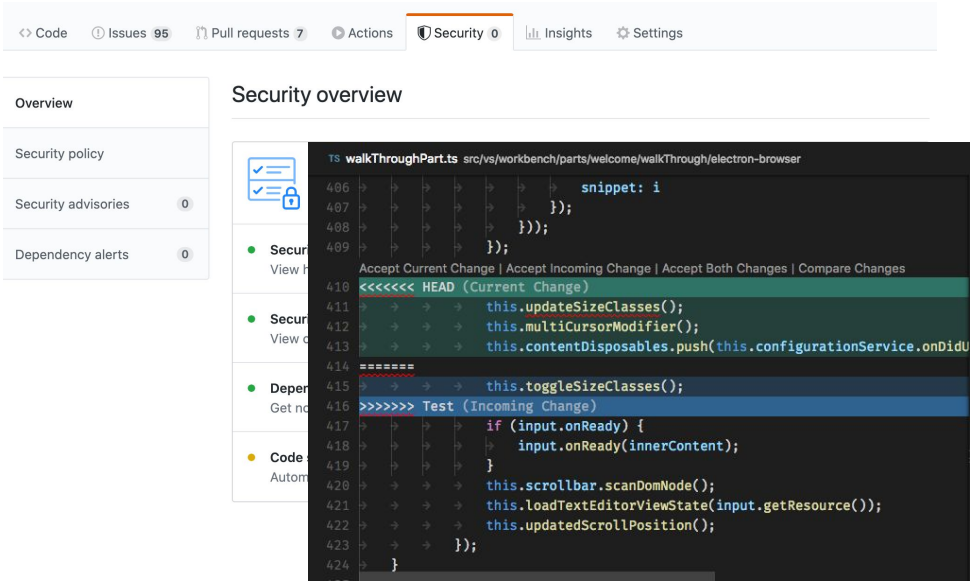
`error`

If `true` will exit `1` which will break the build.

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
```

Security + Developers = ❤️



Semgrep Core Features Coming Soon

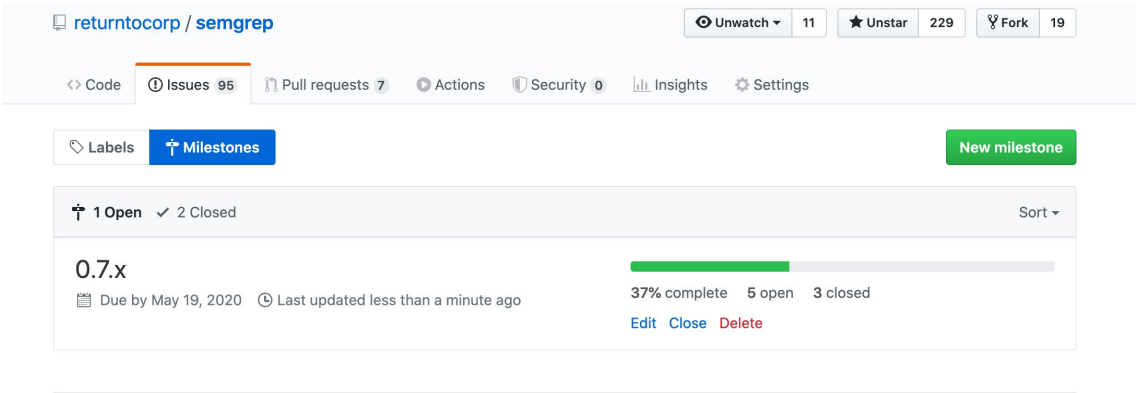
More code equivalences!

```
$X + $Y <==> $Y + $X
```

Integrate typing information

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

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



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**
2. `brew install returntocorp/semgrep/semgrep`
3. `docker run --rm -v "${PWD}:/home/repo" returntocorp/semgrep`

Drew Dennison | @drewdennison | drew@r2c.dev

r2c.dev | @r2cdev

slides are posted at bit.ly/semgrep-toronto