



# Enforcing Code & Security Standards with Semgrep

Grayson Hardaway | grayson@r2c.dev

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

# 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

# Semgrep Trophy Case

CVEs			
CVE	Semgrep rule	Affected software	Description
CVE-2019-5479	<a href="#">javascript.lang.security.detect-non-literal-require</a>	larbitbase-api < v0.5.5	An unintended require vulnerability in <v0.5.5 larbitbase-api may allow an attacker to load arbitrary non-production code (JavaScript file).
CVE-2020-8128	<a href="#">javascript.lang.security.detect-non-literal-require</a>	jsreport < 2.5.0	An unintended require and server-side request forgery vulnerabilities in jsreport version 2.5.0 and earlier allow attackers to execute arbitrary code.
CVE-2020-8129	<a href="#">javascript.lang.security.detect-non-literal-require</a>	script-manager < 0.8.6	An unintended require vulnerability in script-manager npm package version 0.8.6 and earlier may allow attackers to execute arbitrary code.
CVE-2020-7739	<a href="#">javascript.phantom.security.audit.phantom-injection</a>	phantomjs-seo	This affects all versions of package phantomjs-seo. It is possible for an attacker to craft a url that will be passed to a PhantomJS instance allowing for an SSRF attack.

# whois?

me:

Grayson Hardaway, security engineer @ r2c  
Formerly: U.S. Department of Defense



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. Learn Semgrep! 
4. Integration into CI/CD 
5. Get started with Rulesets 

[returntocorp / semgrep](#)

Used by 2 Unwatch 16 Unstar 774 Fork 30

[Code](#) Issues 116 Pull requests 5 Actions Security 0 Insights Settings

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

Edit

static-analysis ocaml grep-like semgrep metavariables Manage topics

756 commits 25 branches 0 packages 50 releases 17 contributors LGPL-2.1

Branch: [develop](#) New pull request Create new file Upload files Find file Clone or download

mschwager Refactor user-defined exceptions (#904) ...	Latest commit b4b3a87 17 hours ago
.bento bento: stop ignoring unused vars (#837)	9 days ago
.github Add benchmark job for PRs (#889)	2 days ago
.vscode 0.44.0 (#138)	4 months ago
docs Merge pull request #897 from returntocorp/release-0.9.0	20 hours ago
install-scripts Add benchmark job for PRs (#889)	2 days ago

[github.com/returntocorp/semgrep](https://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 Padoleau ([@aryx](#)), joined r2c last year. Yoann was the first static analysis hire at Facebook and previously PhD @ Inria, contributor to [coccinelle.lip6.fr](#)

# Language Support

Language	Status
Go	GA ⓘ
Java	GA ⓘ
JavaScript	GA ⓘ
JSON	GA ⓘ
Python	GA ⓘ
Ruby	beta ⓘ
TypeScript	beta ⓘ
JSX	beta ⓘ
TSX	beta ⓘ
OCaml	alpha ⓘ
PHP	alpha ⓘ
C	alpha ⓘ

# grep and Abstract Syntax Trees (ASTs)

# grep, ASTs, and Semgrep

```
exec("ls")  
  
exec(some_var)  
  
exec(arg)  
  
exec(  
    bar  
)  
  
other_exec(foo)  
  
// exec(foo)  
  
print("exec(bar) ")
```

✓ Easy - exec\()

✓ Easy - exec\()

⚠ Handle whitespace exec\s\*\()

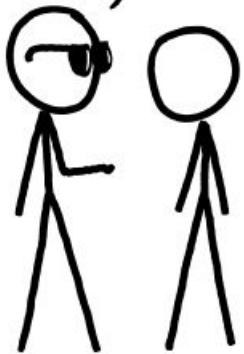
⚠ 😄 Handle whitespace/newlines

🚫 😄 😄 Method suffix matches exec

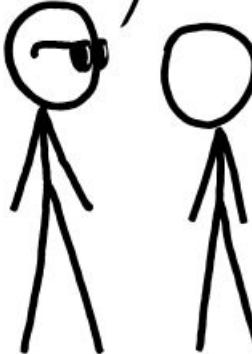
🚫 😄 😄 Is this a comment?

🚫 😄 😄 Is this a string literal?

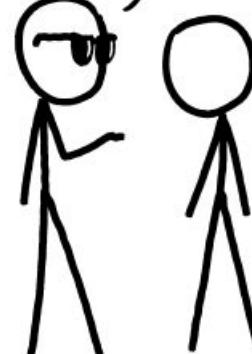
IF YOU'RE HAVIN' PERL  
PROBLEMS I FEEL  
BAD FOR YOU, SON-



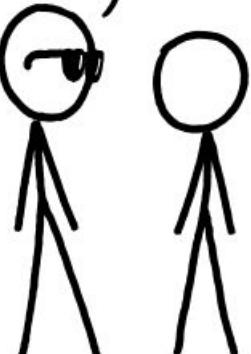
I GOT 99  
PROBLEMS,



SO I USED  
REGULAR  
EXPRESSIONS.



NOW I HAVE  
100 PROBLEMS.



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



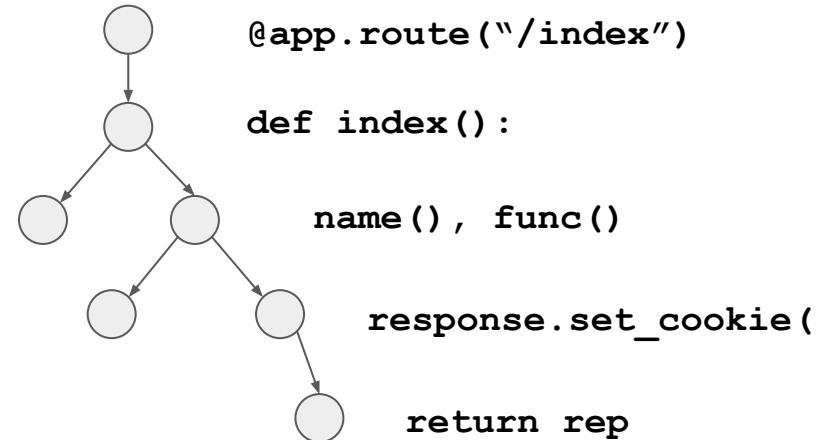
**string**

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

**!=**



**tree**



# Tree Matching



- Many tree matching tools: Bandit, Dlint, ESLint, Flake8, Golang, Gosec, 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)
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;
44   }
45 }
```

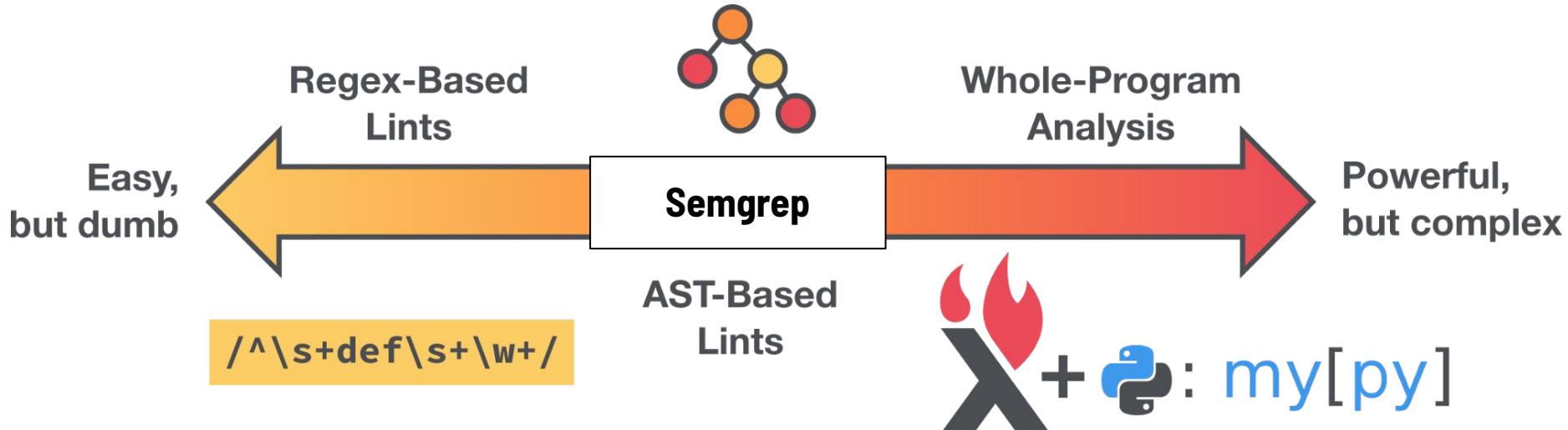
<https://github.com/eslint/eslint/blob/master/lib/rules/no-eval.js>

# Static Analysis at Scale: An Instagram Story



Benjamin Woodruff [Follow](#)

Aug 15, 2019 · 13 min read



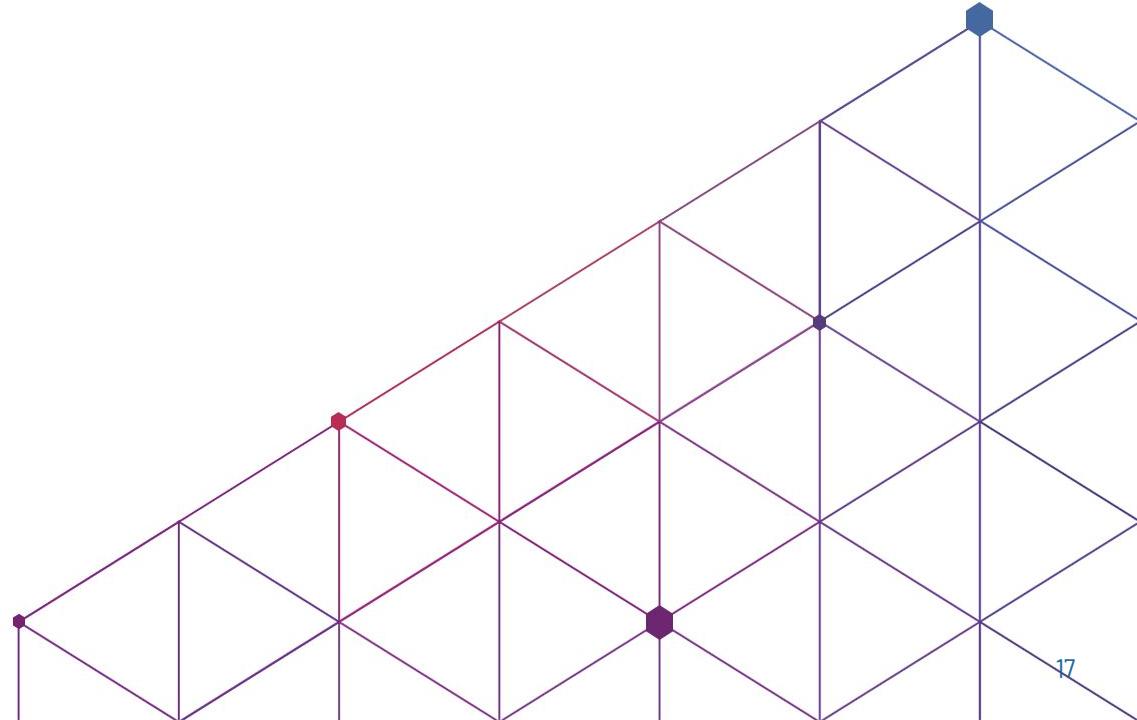
<https://instagram-engineering.com/static-analysis-at-scale-an-instagram-story-8f498ab71a0c>

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/>

# Tutorials

1. Ellipsis ("...") operator 
2. Metavariables
3. Composing Patterns
4. Advanced Features



# Finding Banned, Deprecated, or Dangerous Functions

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

⇒ <https://semgrep.live/EwOP>

Full Solution: <https://semgrep.live/7KGk>

# Hard-coded Secrets, Constant String Arguments

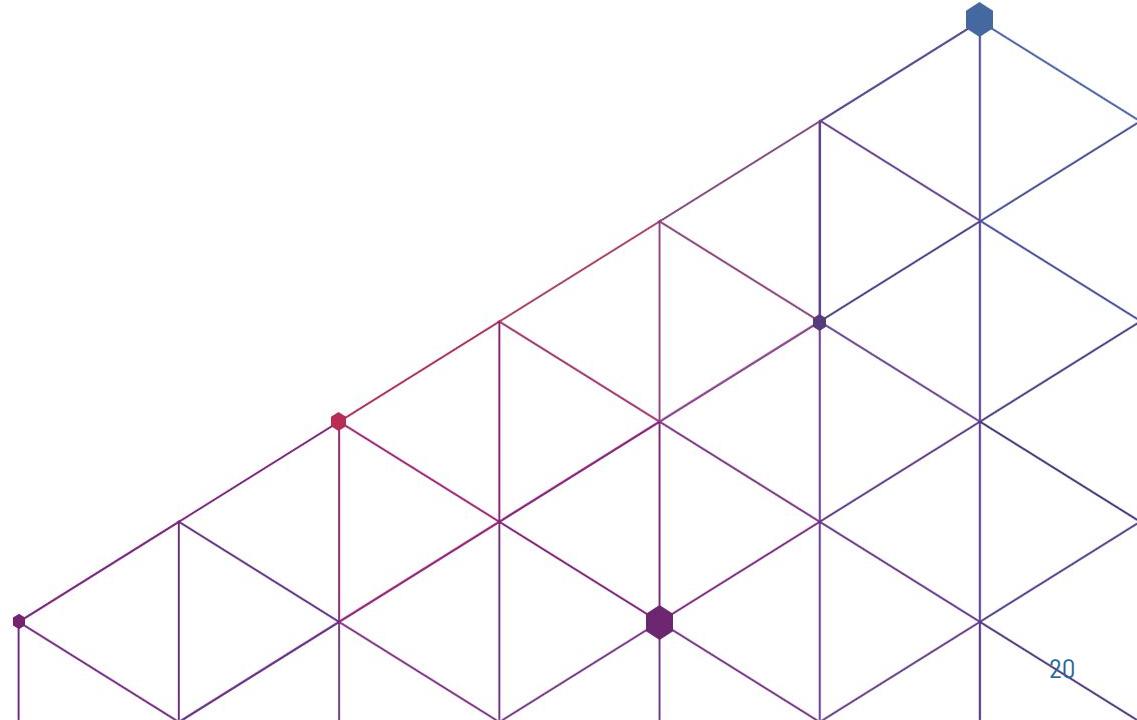
```
s3 = boto3.client(  
    "s3",  
    aws_secret_access_key = "abcd...",  
    aws_access_key_id = "AKIA...")
```

⇒ <https://semgrep.live/RG08/>

Full Solution: <https://semgrep.live/A89w/>

# Tutorials

1. Ellipsis ("...") operator
2. Metavariables 
3. Composing Patterns
4. Advanced Features



# Matching Comparisons with Metavariables

```
5 == 5
7 == 8
if "cat" == "cat":
    print("Yep, they're cats")
```

<https://semgrep.live/61o>

Full Solution: <https://semgrep.live/oB9>

# Send File

(Approximate Data Flow)

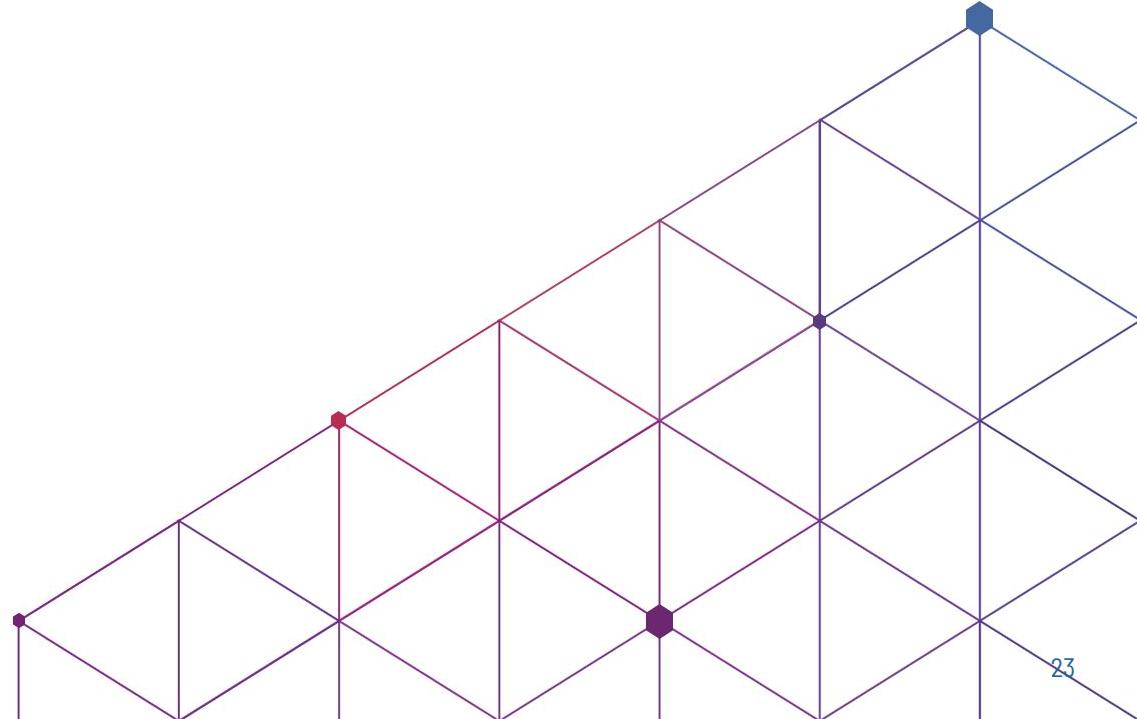
```
@app.route("/get_file/<filename>")  
def get_file(filename):  
    logger.info(f"Getting file {filename}")  
    return flask.send_file(filename,  
                           as_attachment=True)
```

<https://semgrep.live/7dv>

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

# Tutorials

1. Ellipsis ("...") operator
2. Metavariables
3. Composing Patterns 
4. Advanced Features



# Finding Insecure SSL Configurations (Composing patterns)

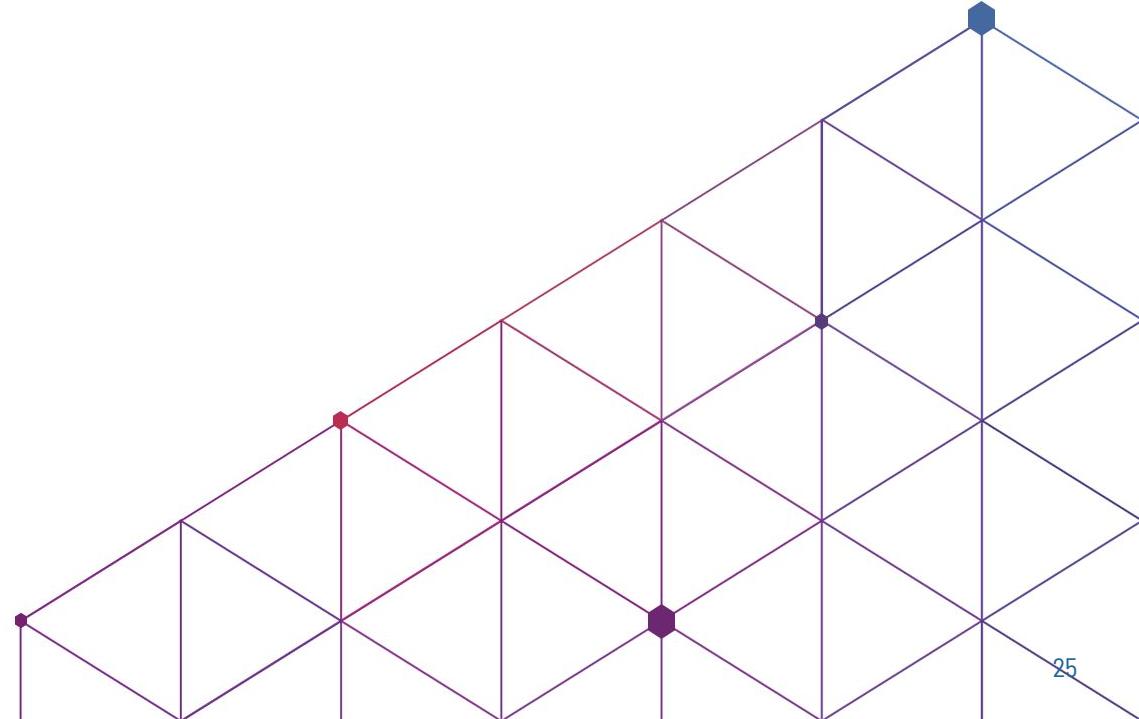
```
&tls.Config{  
    KeyLogWriter: w,  
    MinVersion: tls.VersionSSL30,  
    Rand: randSource{},  
    InsecureSkipVerify: true,  
}
```

⇒ <https://semgrep.live/DbYd>



# Tutorials

1. Ellipsis ("...") operator
2. Metavariables
3. Composing Patterns
4. Advanced Features 🧑‍🔧



# Order of API Calls Must be Enforced (Business Logic)

```
/*
 * 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>

# Detect HTTP

## (Inline String RegExes)

```
func bad1() {  
    req, err := http.NewRequest("GET", "http://example.com", nil)  
}
```

```
pattern: |  
    http.NewRequest(...., "=~/[hH][tT][tT][pP]://.*/", ....)
```

<https://semgrep.dev/editor?registry=problem-based-packs.insecure-transport.go-stdlib.http-customized-request>

## Automatically fix an insecure SSL configuration (Autofix)

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

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

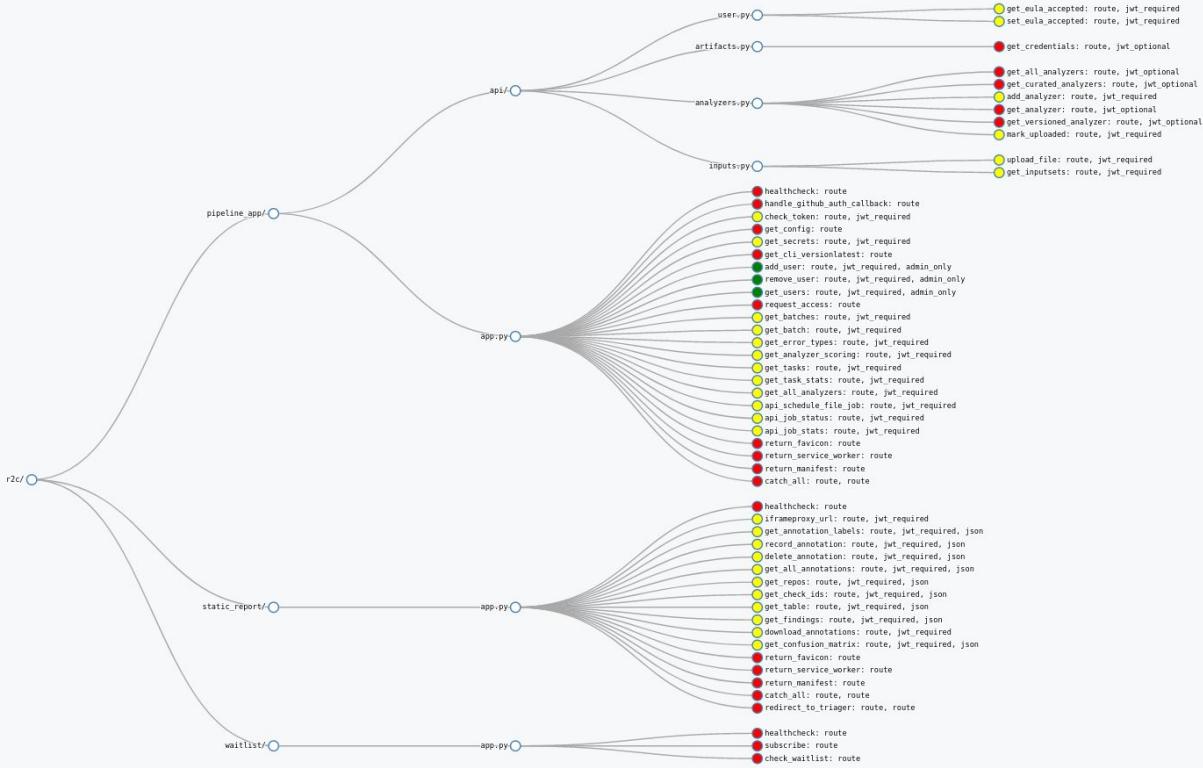
# Identify API endpoints in Java Spring (Message Interpolation)

```
@RequestMapping(method = RequestMethod.GET)
@Authorize(Permissions.ADMIN)
@ResponseBody
public ResponseEntity<Map<String, Object>> list() {
    return new ResponseEntity<>(result, HttpStatus.OK);
}
```

<https://semgrep.live/clintgibler:spring-routes-try>

Full Solution: <https://semgrep.live/clintgibler:spring-routes>

# Visualizing API endpoints



# Terraform

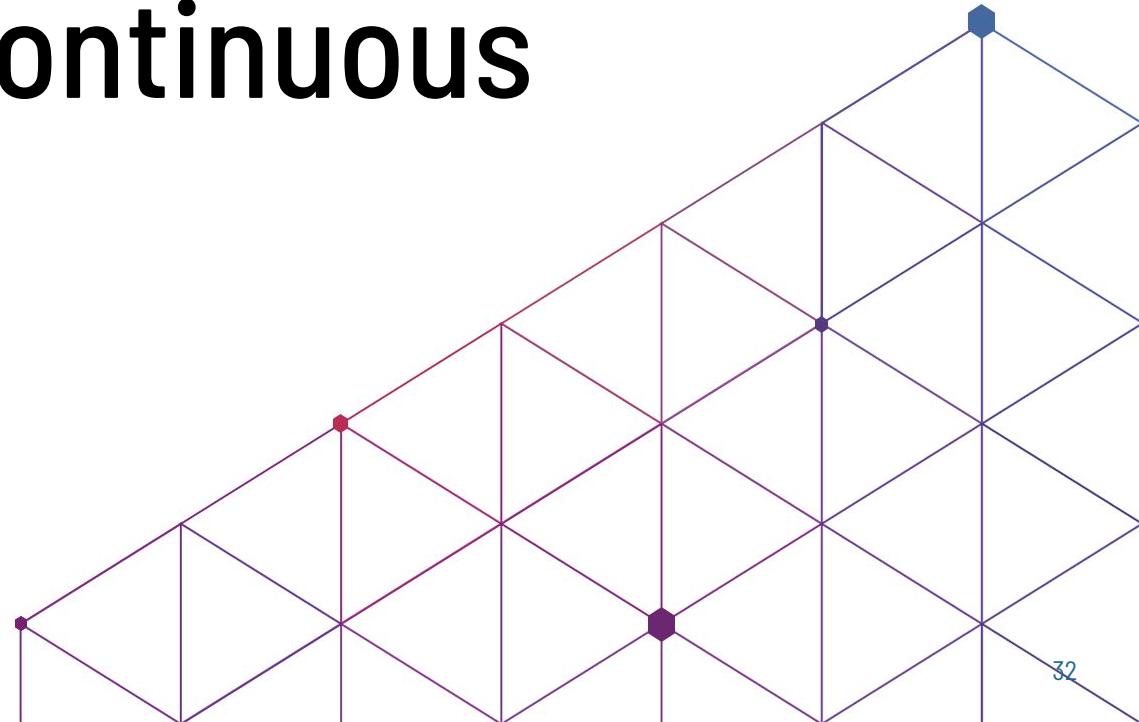
## (Generic Language Support)

```
resource "aws_s3_bucket" "b" {  
    bucket = "my-tf-test-bucket"  
    acl     = "public-read-write"  
    ...  
}
```

```
pattern: |  
    acl = "public-read-write"
```

<https://semgrep.dev/s/ne0Z/>

# Guard with Continuous Integration



# 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

gitpre-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
```

✓ Linters  
on: pull\_request

✓ super-linter

✓ pre-commit

✗ semgrep with managed policy

Linters / semgrep with managed policy

failed 1 hour ago in 1m 25s

Search logs

► ✓ Set up job

0s

► ✓ Pull returntacorp/semgrep-action:v1

5s

► ✓ Run actions/checkout@v1

2s

▼ ✗ Run returntacorp/semgrep-action@v1

1m 18s

```
GITHUB_EVENT_NAME -e GITHUB_SERVER_URL -e GITHUB_API_URL -e GITHUB_GRAPHQL_URL -e GITHUB_WORKSPACE -e GITHUB_ACTION -e GITHUB_EVENT_PATH -e RUNNER_OS -e RUNNER_TOOL_CACHE -e RUNNER_TEMP -e RUNNER_WORKSPACE
-e ACTIONS_RUNTIME_URL -e ACTIONS_RUNTIME_TOKEN -e ACTIONS_CACHE_URL -e GITHUB_ACTIONS=true -e CI=true -v "/var/run/docker.sock":"/var/run/docker.sock" -v
"/home/runner/work/_temp/_github_home":"/github/home" -v "/home/runner/work/_temp/_github_workflow":"/github/workflow" -v
"/home/runner/work/_temp/_github_workflow/_actions":"/github/workspace"
returntacorp/semgrep-action:v1

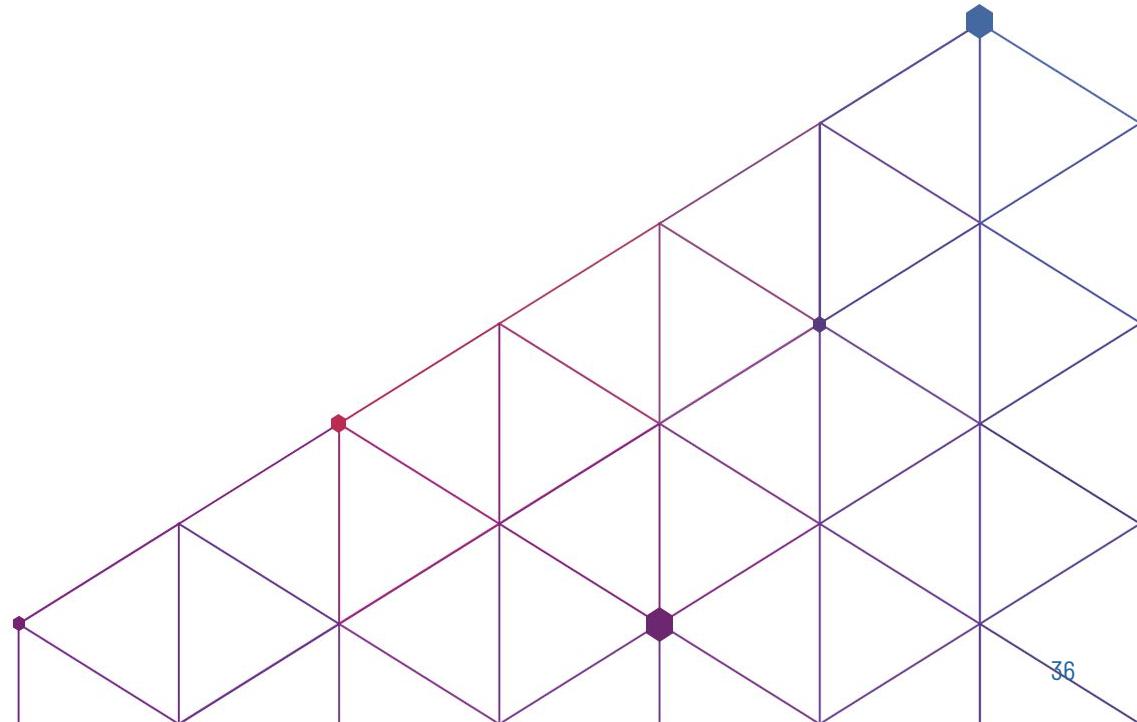
6 === detecting environment
7 | versions      - semgrep 0.17.0 on Python 3.8.5
8 | environment   - running in github-actions, triggering event is 'pull_request'
9 | semgrep.dev   - logged in as deployment #1
10 === setting up agent configuration
11 | using semgrep rules configured on the web UI
12 | 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

0s

```
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
```

# Registry



# Community rule registry

## Community participation

- > 900 rules by r2c + community
- Community contributors
  - **NodeJSScan**
  - **Damian Gryski, author of Go-Perf-Book**
  - **OWASP Contributors**
  - **You?**
- Organized into sets
  - XSS
  - JWT
  - Best practices
  - etc.

### Getting Started

These rulesets cover a wide range of use cases. Start here to get up and running quickly.

#### r2c-ci



Scan for runtime errors, logic bugs, and high-confidence security vulnerabilities. Recommended for use in CI to block serious issues from reaching...

[Go](#) [Java](#) [JavaScript](#) [Python](#) [Ruby](#)

#### r2c-security-audit



Scan code for potential security issues that require additional review. Recommended for teams looking to set up guardrails or to flag troublesome spots for...

[Ruby](#) [JavaScript](#) [Go](#) [Java](#) [C](#)

### Enforce Secure Guardrails

Use Semgrep to ensure your code enforces secure defaults and framework protections, which can proactively eradicate entire classes of vulnerabilities. Avoid playing bug whack-a-mole and scale your security program.

#### insecure-transport



Ensure your code communicates over encrypted channels instead of plaintext.

[Java](#) [JavaScript](#) [Go](#)

#### jwt



Avoid common JWT security mistakes

[Go](#) [Ruby](#) [Python](#) [Java](#)  
[JavaScript](#) [TypeScript](#)

#### xss



Secure defaults for XSS prevention across 5 different languages

[Go](#) [Ruby](#) [Python](#) [Java](#)  
[JavaScript](#)



# Community rule registry

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

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

← → ⌂ semgrep.dev/packs

Apps Projects · returnto... XSS Payloads Security Reading OS X Screencast t... The npm Blog — n... Malware Checks...

Star 1,649 ⚡ 🐾 🐣 🐛 🐞

Semgrep Live Editor Tutorial Registry Docs

Packs All Rules My Created Packs My Used Packs

A pack is a collection of Semgrep rules. You can find all available pre-written rules on the 'All Rules' tab. You can also create your own rules using the live editor. Click one of these packs to find the `--config` argument needed to run it locally.

### Featured Packs

**bandit**  
Selected rules from Bandit, a security checker for Python, implemented in Semgrep.

python security bandit injection  
deserialization xss crypto owasp

**eslint-plugin-security**  
Selected rules from eslint-plugin-security, a security plugin for ESLint, implemented in Semgrep.

eslint security

**findsecbugs**  
Selected rules from FindSecBugs, a security checker for Java, implemented in Semgrep.

java security findsecbugs xxe  
deserialization owasp xss  
injection

**gosec**  
Selected rules from gosec, a security checker for Golang, implemented in Semgrep.

go golang security gosec xss  
zip net crypto

**nodejscan**  
Security rules for Node.js

node nodejs javascript security  
xss owasp injection jwt

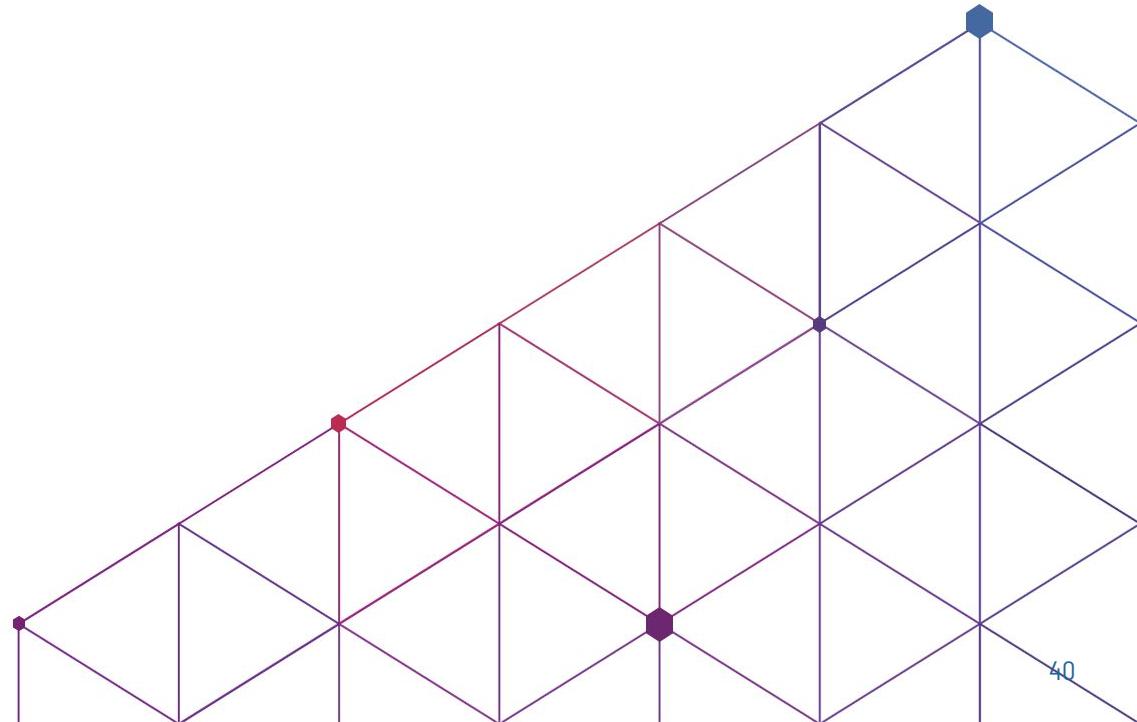
**r2c-CI**  
Scan for runtime errors, logic bugs, and high-confidence security vulnerabilities.  
Recommended for use in CI to catch security issues early.

CI cookies correctness crypto  
csrf go injection java javascript  
python security spring xss xxe  
logic logic bugs runtime errors  
slower

\$ semgrep --config=https://semgrep.dev/p/gosec

39

# Summary



# Integration points

- Editor / IDE ([VS Code Extension](#) in beta)
- Git pre-commit hook
- CI/CD pipeline
- If using managed Semgrep on semgrep.dev, rules can either **block** or **notify**

# Getting started with rules

- Choose existing rule sets from semgrep.dev

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

- Create your own rule set from existing rules
- Write your own!

python.requests.security

Run Locally Add to Policy ▾

**no-auth-over-http**

Example

```
# ok:no-auth-over-http
good_url = "https://www.github.com"
bad_url = "http://www.github.com"

# ruleid:no-auth-over-http
r = requests.post("http://www.github.com", auth=('user', 'pass'))

# ok:no-auth-over-http
r = requests.post(good_url, auth=('user', 'pass'))

# ok:no-auth-over-http
```

Open

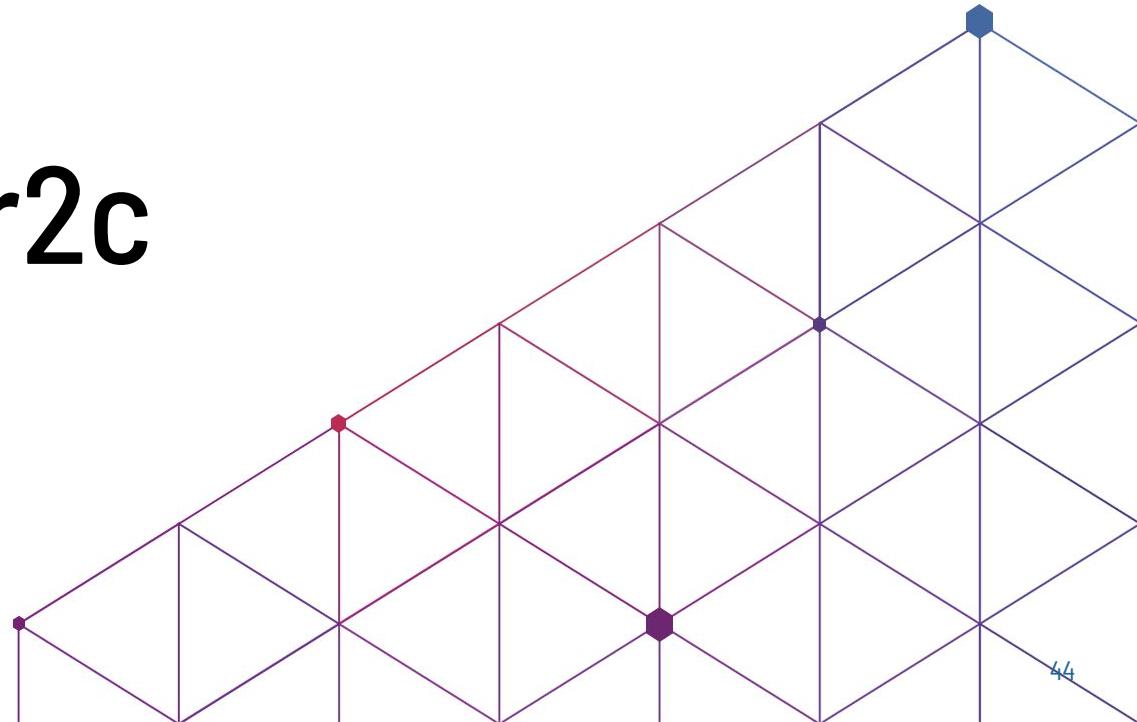
Authentication detected over HTTP. HTTP does not provide any encryption or protection for these authentication credentials. This may expose these credentials to unauthorized parties. Use 'https://' instead.



Set your **security policy in stone** with  
**automated scanning**.

<https://r2c.dev/blog/2020/fixing-leaky-logs-how-to-find-a-bug-and-ensure-it-never-returns/>

# Case Study: r2c



```
logging.getLogger("sqlalchemy.engine.base.Engine").setLevel(logging.INFO)
```

This configures SQLAlchemy to log all SQL statements, together with passed parameters. Let's look at some of the output we saw:

```
INFO:werkzeug:127.0.0.1 - - [25/Sep/2020 11:50:01] "POST /api/auth/authenticate"
INFO:sqlalchemy.engine.base.Engine:BEGIN (implicit)
INFO:sqlalchemy.engine.base.Engine:SELECT token.id AS token_id, token.token AS
FROM token
WHERE token.token = %(token_1)s
LIMIT %(param_1)s
INFO:sqlalchemy.engine.base.Engine:{'token_1': '$2a$10$KVsyW1jjKn.pvkVi3w9Rn.1mw
```

...Uh-oh.

```
class ObfuscatedString(types.TypeDecorator):
    """
        String column type for use with SQLAlchemy models whose
        content should not appear in logs or exceptions
    """

    impl = types.String

    class Repr(str):
        def __repr__(self) -> str:
            return "*****"

    def process_bind_param(self, value: Optional[str], dialect: Any) -> Optional[str]:
        return self.Repr(value) if value else None

    def process_result_value(
        self, value: Optional[Repr], dialect: Any
    ) -> Optional[str]:
        return str(value) if value else None

setattr(db, "ObfuscatedString", ObfuscatedString)
```

```
class Token(db.Model):
    ...
    token = db.Column(db.ObfuscatedString, ...)
    ...
```

We then re-enabled INFO logging, and checked that we were properly obfuscating text:

```
INFO:werkzeug:127.0.0.1 - - [25/Sep/2020 13:48:55] "GET /api/agent/deployments/
INFO:sqlalchemy.engine.base.Engine:BEGIN (implicit)
INFO:sqlalchemy.engine.base.Engine:SELECT token.id AS token_id, token.token AS
FROM token
WHERE token.token = %(token_1)s
LIMIT %(param_1)s
INFO:sqlalchemy.engine.base.Engine:{'token_1': '*****', 'param_1': 1}
```

For completeness, we also validated in our development database console that the correct values were stored and retrieved.

Great success! 🚢 Ship it.

```
rules:
- id: obfuscate-sensitive-string-columns
  patterns:
    - pattern: |
        $COLUMN = db.Column(db.String, ...)
    - metavariable-regex:
        metavariable: $COLUMN
        regex: '.*(?:<![A-Za-z])(token|key|email|secret)(?:![A-RT-Za-rt-z]).*'
  message: |
    '$COLUMN' may expose sensitive information in logs and exceptions. Use
    'db.ObfuscatedString' instead of 'db.String'.
  severity: WARNING
```

# Coming Soon

Improved language support, new languages  
(Kotlin on deck)

More rules + prevention cheatsheets

Semgrep Community

Centrally manage Semgrep on your repos!

VS Code extension (in beta!)

"Generic" patterns

The screenshot shows the Semgrep web interface. At the top, there's a navigation bar with 'Semgrep' logo, 'Write', 'Explore', 'Manage', and 'Docs'. On the right, it shows the user 'returntocorp' and a profile icon.

The main area has a sidebar with 'Projects' (selected), 'Policies' (highlighted in blue), 'Notifications', and 'Settings'.

Below the sidebar, there are four policy cards:

- bandit**: Items used on dormbase/dormbase
- Python Packages**: Items used on palets/flask semgrep
- default**: Items used on returntocorp/infrastructure, returntocorp/cli and 5 more...
- policy one**: Items used on daghan/lets-be-bad-guys

A modal window titled 'Web Apps' is open, showing items used on returntocorp/semgrep-app, returntocorp/echelon-backend, and 4 more... It includes a 'Delete Policy' button and a 'Download YAML' button.

At the bottom, there's a table for managing rulesets:

	RULESET	SNIPPET	YAML	JSON
bandit	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
eslint-plugin-security	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
nodejsscan	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
r2c-security-audit	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
mschwager:harcoded-rsa-private-key	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>



# Semgrep

lightweight static analysis for many languages

Locally:

1. `brew install semgrep`
2. `pip install semgrep`

Online editor:

[semgrep.live](https://semgrep.live)





# Semgrep

lightweight static analysis for many languages



Grayson Hardaway | grayson@r2c.dev

[r2c.dev](https://r2c.dev)

| [@r2cdev](https://twitter.com/r2cdev) | [r2c Community Slack](https://r2c.slack.com)



<https://r2c.dev/survey> ← plz :)

