# Enforcing Code & Security Standards with Semgrep

Clara McCreery | clara@returntocorp.com

@r2cdev

# tl;dr – This Talk

- Secure code is hard
- Static analysis tools are too noisy / too slow
- Grep is faster but 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

# Outline

1. Background 🏞️
2. `grep` and Abstract Syntax Trees (ASTs) 🌲
3. Learn Semgrep! 🎓
4. Integration into CI/CD ⚙️
5. Get started with Rulesets 📖

# Outline

1. **Background** 🏞️
2. `grep` and Abstract Syntax Trees (ASTs) 🌲
3. Learn Semgrep! 🎓
4. Integration into CI/CD ⚙️
5. Get started with Rulesets 📖

# Who is this?

**me:**

Clara McCreery, software engineer @ r2c

MS in Computer Science from Stanford University

Formerly: researcher at MIT Lincoln Laboratory

**r2c:**

We're a static analysis startup in San Francisco on a mission to profoundly improve software security and reliability

# Semgrep, Est. 2009

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

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

returntocorp / semgrep

Watch 46 | Star 2.3k | Fork 97

Code | Issues 183 | Pull requests 4 | Actions | Security | Insights

develop | 28 branches | 85 tags | Go to file | Code

underyx Update Code of Conduct to Contributor Covenant v2 (#2126) 1b15418 2 hours ago 1,557 commits

| | | |
|---|---|---|
| .circleci | Remove semgrep-core/finding/, and enable back LSP (#1921) | last month |
| .github | ci: check access to secrets before pushing (#2139) | 2 hours ago |
| .vscode | add pre-commit | 10 months ago |
| ocaml-tree-sitter @ 5807c82 | Ignore tree-sitter error nodes (#1876) | last month |
| scripts | changing ext type for ocaml (#2116) | 7 days ago |
| semgrep-core | adding tests for metavariables in class definitions (#2140) | 3 hours ago |
| semgrep | Do not follow symlinks (#2113) | 2 hours ago |
| spacegrep | Add bits about installing spacegrep for development (#2138) | 20 hours ago |
| stubs | Improve jsonschema error when missing top-level key (#1642) | 3 months ago |
| .dockerignore | WIP: Ruby tree-sitter integration (#1127) | 5 months ago |
| .gitignore | Remove many of the .opam in semgrep-core (#1881) | last month |
| .gitmodules | Move pfff under semgrep-core (#1601) | 3 months ago |
| .ocp-indent | minimal typescript support (#1634) | 3 months ago |
| .pre-commit-config.yaml | Bump pre-commit hooks (#2100) | 11 days ago |
| .pre-commit-hooks.yaml | Add python backed pre-commit hook (#1401) | 4 months ago |
| .semgrepignore | Rename .bentoignore to .semgrepignore (#1268) | 5 months ago |
| CHANGELOG.md | Do not follow symlinks (#2113) | 2 hours ago |

**About**

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

semgrep.dev

static-analysis | static-code-analysis | python | java | go | javascript | c | sast

Readme

LGPL-2.1 License

**Releases** 85

Release v0.32.0 Latest
13 days ago

+ 84 releases

**Packages**

No packages published

**Used by** 17

+ 9

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

# 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 |

# Outline

1. Background 🏞️
2. **`grep` and Abstract Syntax Trees (ASTs)** 🎄
3. Learn Semgrep! 🎓
4. Integration into CI/CD ⚙️
5. Get started with Rulesets 📖

# 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*\(`

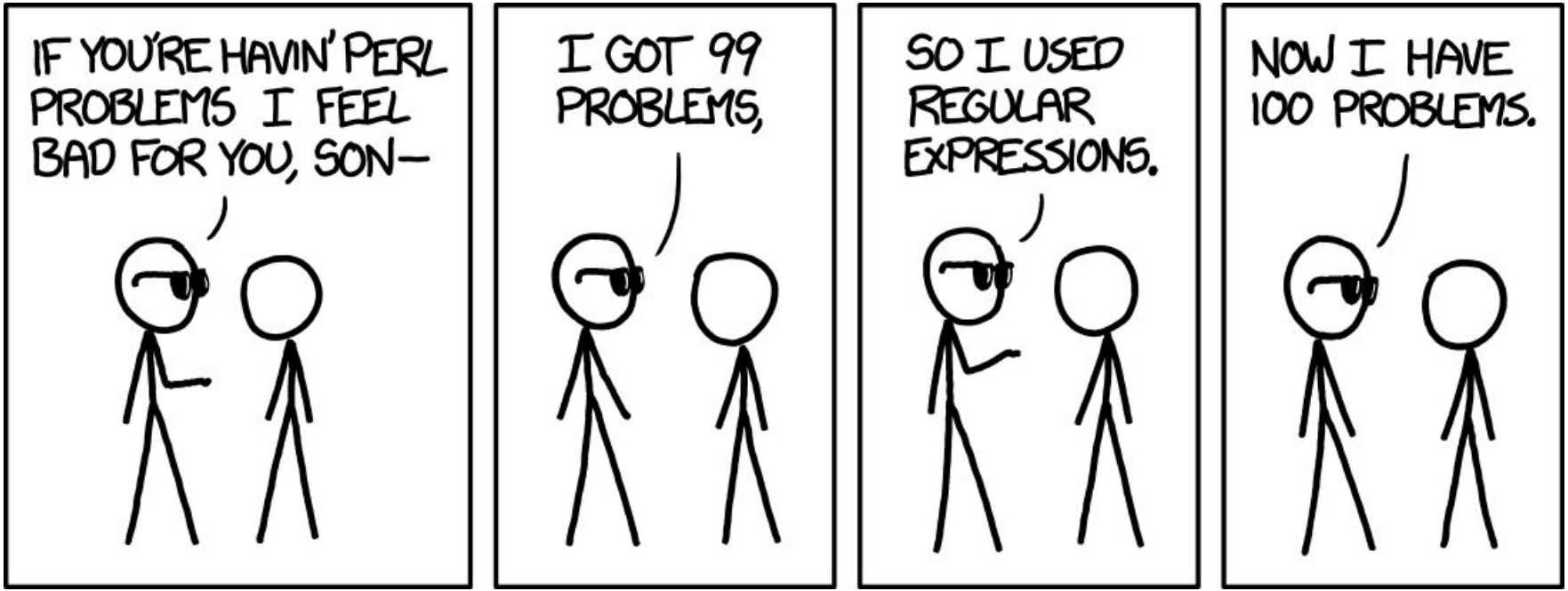
⚠️ 😅 Handle whitespace/newlines

🛑😅😅 Method suffix matches `exec`

🛑😅😅 Is this a comment?

🛑😅😅 Is this a string literal?

# 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.

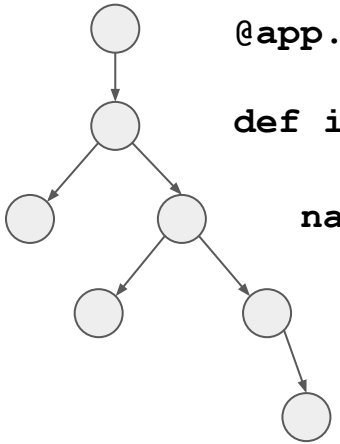
# Code is not a string, it's a tree

**string**

!=

**tree**

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



```
@app.route("/index")

def index():

    name(), func()

        response.set_cookie(

            return rep
```

# Tree Matching 🌲

- Many tree matching tools: Bandit, Dlint, ESLint, Flake8, Golint, 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) 183e300 on Mar 17

20 contributors

307 lines (258 sloc) | 9.24 KB

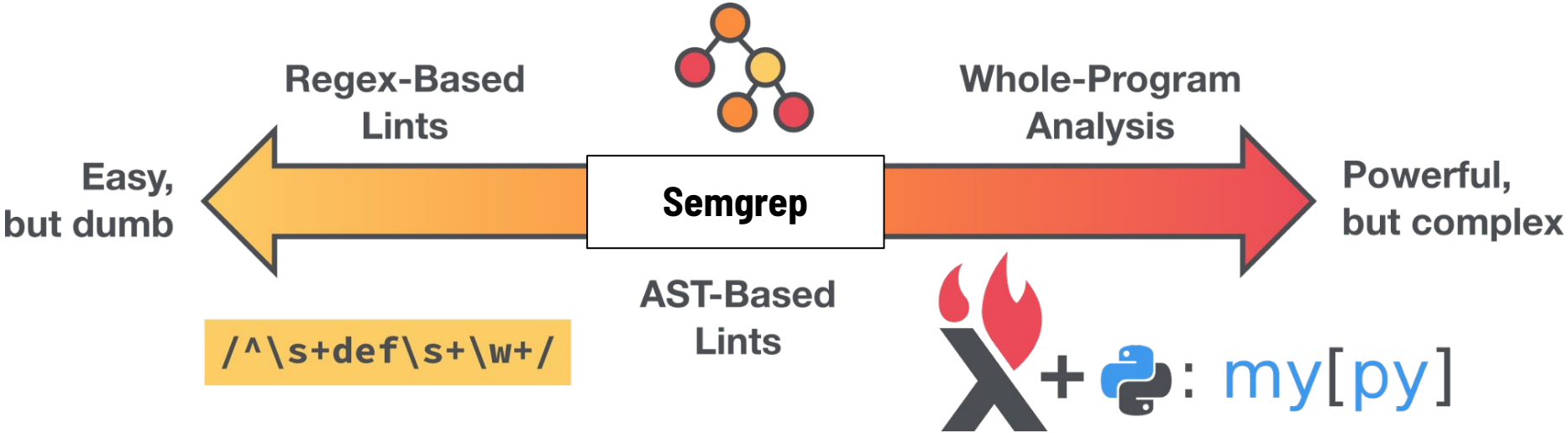
Raw Blame History

```
/**
 * @fileoverview Rule to flag use of eval() statement
 * @author Nicholas C. Zakas
 */

"use strict";

//------------------------------------------------------------------------------
// Requirements
//------------------------------------------------------------------------------

const astUtils = require("./utils/ast-utils");

//------------------------------------------------------------------------------
// Helpers
//------------------------------------------------------------------------------

const candidatesOfGlobalObject = Object.freeze([
    "global",
    "window",
    "globalThis"
]);

/**
 * Checks a given node is a Identifier node of the specified name.
 * @param {ASTNode} node A node to check.
 * @param {string} name A name to check.
 * @returns {boolean} `true` if the node is a Identifier node of the name.
 */
function isIdentifier(node, name) {
    return node.type === "Identifier" && node.name === name;
}

/**
 * Checks a given node is a Literal node of the specified string value.
 * @param {ASTNode} node A node to check.
 * @param {string} name A name to check.
 * @returns {boolean} `true` if the node is a Literal node of the name.
 */
function isConstant(node, name) {
    switch (node.type) {
        case "Literal":
```

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**.

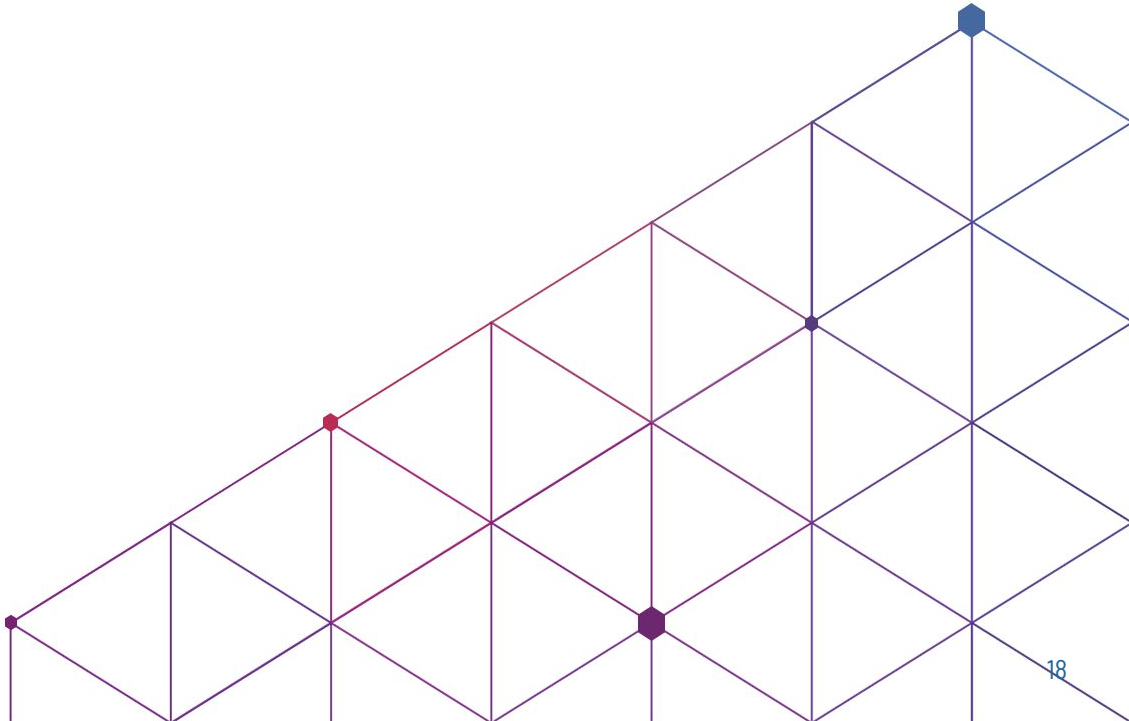
https://r2c.dev/blog/2020/why-i-moved-to-semgrep-for-all-my-code-analysis/

# Outline

1. Background 🏞️
2. `grep` and Abstract Syntax Trees (ASTs) 🌲
3. **Learn Semgrep!** 🎓
4. Integration into CI/CD ⚙️
5. Get started with Rulesets 📖

# Tutorials

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



# Finding Banned, Deprecated, or Dangerous Functions

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

⇒ https://semgrep.dev/s/4B2g/

**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)

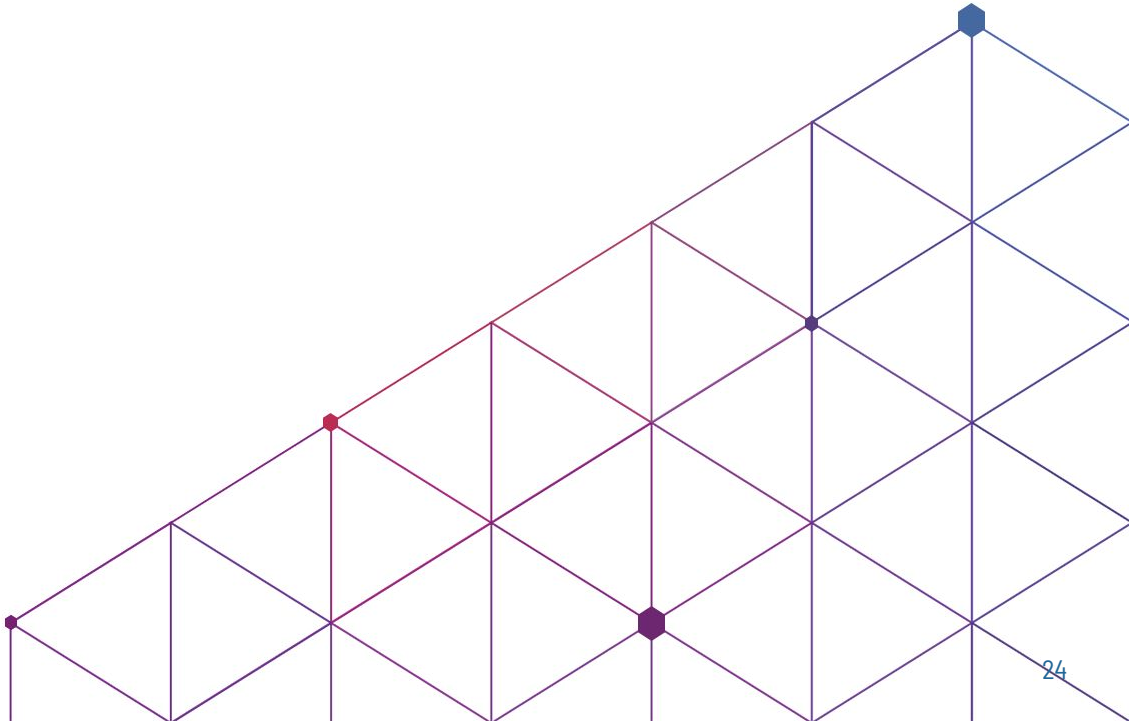
```
@app.route("/get_file/<filename>")
def get_file(filename):
  logger.info(f"Getting file {filename}")
  return flask.send_file(filename,
     as_attachment=True)
```

https://semgrep.live/7dv

**Solution:** https://semgrep.live/EN8

# Tutorials

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



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

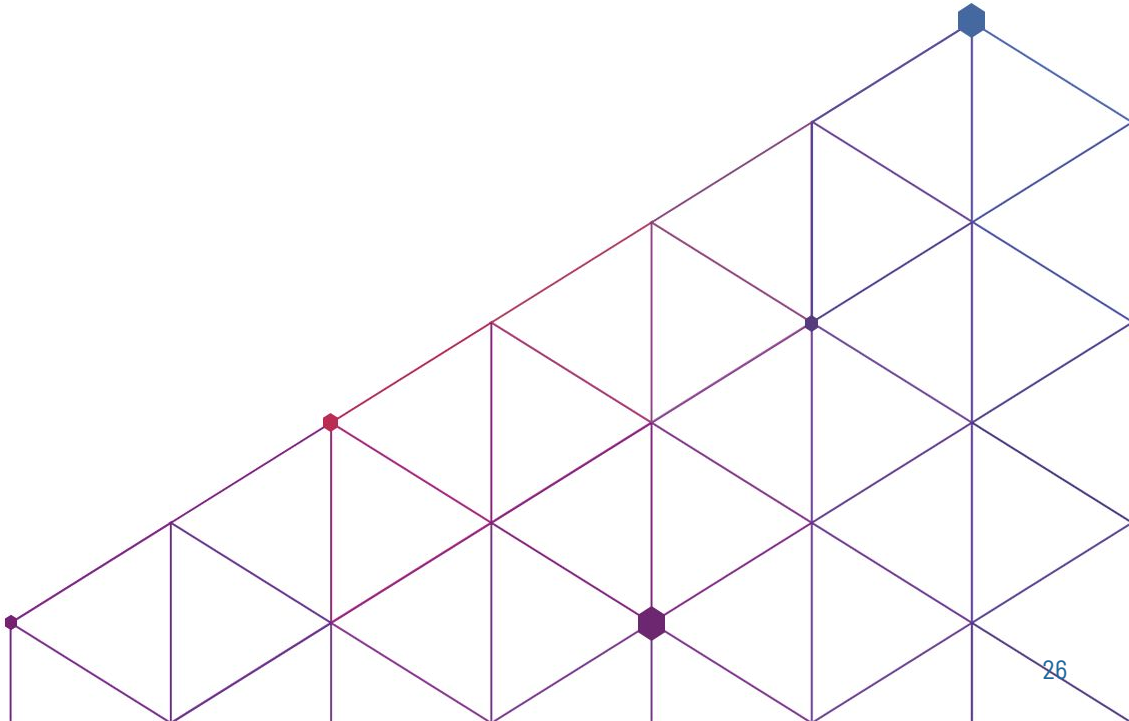
```
/*
* In this financial trading application, every transaction
* MUST be verified before it is made
*
* Specifically:verify_transaction() must be called on a transaction
* object before that object is passed to make_transaction()
*/
```

https://semgrep.live/6JqL

**Full Solution:** https://semgrep.live/oqZ6

# Tutorials

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



# Detect HTTP (Autofix + 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

# 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/

# Outline

1. Background 🏞️
2. `grep` and Abstract Syntax Trees (ASTs) 🌲
3. Learn Semgrep! 🎓
4. **Integration into CI/CD** ⚙️
5. Get started with Rulesets 📖

# 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
```

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 returntocorp/semgrep-action:v1 5s
- Run actions/checkout@v1 2s
- Run returntocorp/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 [redacted]:"/github/workspace"
returntocorp/semgrep-action:v1
=== detecting environment
| versions     - semgrep 0.17.0 on Python 3.8.5
| environment  - running in github-actions, triggering event is 'pull_request'
| semgrep.dev - logged in as deployment #1
=== setting up agent configuration
| using semgrep rules configured on the web UI
| using default path ignore rules of common test and dependency directories
| adding further path ignore rules configured on the web UI
| looking at 1 changed path
| found 1 file in the paths to be scanned
=== looking for current issues in 1 file
| 1 current issue found
=== looking for pre-existing issues in 1 file
| 1 pre-existing issue found
python.flask.security.injection.path-traversal-open.path-traversal-open
        [redacted].py:459

      ┆
  459┆   open(path).readlines(), mimetype="text/plain"
      ┆
      = Found request data in a call to 'open'. Ensure the request data is
        validated or sanitized, otherwise it could result in path traversal
        attacks.

=== exiting with failing status
```

- Complete job 0s

```
=== setting up agent configuration
| using semgrep rules configured on the web UI
| using default path ignore rules of common test and dependency directories
| adding further path ignore rules configured on the web UI
| looking at 1 changed path
| found 1 file in the paths to be scanned
=== looking for current issues in 1 file
| 1 current issue found
=== looking for pre-existing issues in 1 file
| 1 pre-existing issue found
python.flask.security.injection.path-traversal-open.path-traversal-open
     [redacted].py:459
      |
  459|   open(path).readlines(), mimetype="text/plain"
      |
      = Found request data in a call to 'open'. Ensure the request data is
        validated or sanitized, otherwise it could result in path traversal
        attacks.

=== exiting with failing status
```

Get started at https://semgrep.dev/manage

# Outline

1. Background 🏞️
2. `grep` and Abstract Syntax Trees (ASTs) 🌲
3. Learn Semgrep! 🎓
4. Integration into CI/CD ⚙️
5. **Get started with Rulesets 📖**

# Community rule registry

https://semgrep.dev/explore

## 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**

by r2c

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**

by r2c

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**

by Colleen Dai

Ensure your code communicates over encrypted channels instead of plaintext.

Java JavaScript Go

**jwt**

by Vasilii Ermilov

Avoid common JWT security mistakes

Go Ruby Python Java JavaScript TypeScript

**xss**

by Grayson Hardaway

Secure defaults for XSS prevention across 5 different languages

Go Ruby Python Java JavaScript

# Community rule registry

semgrep.live/explore ⇒ github.com/returntocorp/**semgrep-rules**

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

Example config url: “https://semgrep.dev/p/r2c-ci” or “just p/r2c-ci”

semgrep.dev/packs

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

Semgrep | Live Editor | Tutorial | Registry | Docs | Star 1,649

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

### nodejsscan

Security rules for Node.js

node | nodejs | javascript | security | xss | owasp | injection | jwt

### r2c-CI

Scan for runtime errors, logic bus, and high-confidence security vulnerabilities. Recommended for use in CI to

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
```

# 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

Open

```
# 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
```

Authentication detected over HTTP. HTTP does not provide any encryption or protection for these authentication credentials. This may expose these credentials to unauthhorized 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.

```python
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) -> Optiona
        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.

```yaml
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
```

# Semgrep

lightweight static analysis for many languages

Locally:

1. `pip install semgrep`
2. `semgrep --config p/r2c-ci`

Online editor:

Semgrep.live

Tutorial:

semgrep.dev/learn

# Semgrep

lightweight static analysis for many languages

# ?

Clara McCreery | clara@returntocorp.com

r2c.dev | @r2cdev | r2c Community Slack

https://**r2c.dev**/survey ← plz :)

# Semgrep Trophy Case

| CVEs | | | |
|---|---|---|---|
| **CVE** | **Semgrep rule** | **Affected software** | **Description** |
| CVE-2019-5479 | javascript.lang.security.detect-non-literal-require | larbitbase-api < v0.5.5 | An unintended require vulnerability in <v0.5.5 larvitbase-api may allow an attacker to load arbitrary non-production code (JavaScript file). |
| CVE-2020-8128 | javascript.lang.security.detect-non-literal-require | 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 | javascript.lang.security.detect-non-literal-require | 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 | javascript.phantom.security.audit.phantom-injection | 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. |