Detect complex code patterns using semantic grep

Bence Nagy | bence@r2c.dev

@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
Use Semgrep to:

- **Search:** Find security bugs
- **Guard:** Enforce best practices
- **Monitor:** Get notifications about new matches
- **Migrate:** Refactor code easily
$ whois
@underyx (Bence Nagy) engineer @ r2c
previously at:
Astroscreen (information warfare)
Kiwi.com (travel)

$ getent group r2c

We’re an SF based code analysis startup.

Mission: profoundly improve code security & reliability
Outline

1. A 60 second history 🏞
2. Trees. 🌲 (well... syntax trees)
3. Learning Semgrep! 🎓
4. Integration into CI/CD 🔄
5. Semgrep Rules Registry 📚
GitHub repository for `returntocorp/semgrep`. The repository has 168 issues, 6 pull requests, and 77 forks. The code section shows a file named `.circledi` with a note to use a new Python rule to detect incorrect usage. A file named `.github` is mentioned, indicating the addition of basic metrics for semgrep-code. Another file named `.vscode` is mentioned, which contains pre-commit messages. The `docs` directory shows release changes 2 days ago, and a file named `ocaml-tree-sitter` uses the latest ocaml-tree-sitter and has recent updates.

About section:

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 Padioleau (@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

- JavaScript
- Go
- Java
- C (beta)
- OCaml (beta)
- Ruby (beta)
- JSON (beta)
- TypeScript (beta)
- PHP (alpha)

License

Branch: develop  semgrep / LICENSE

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.
grep and Abstract Syntax Trees (ASTs)
I got 99 problems,

so I used regular expressions.

Now I have 100 false positives.
Code is not a string, it’s a tree

```python
code

```
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 old `crypto` in 94 LOC 👍

Static Analysis at Scale: An Instagram Story

https://instagram-engineering.com/static-analysis-at-scale-an-instagram-story-8f498ab71a0c
Semgrep: reason about **analysis** like you reason about **code**

write `eval(...) to match eval(request)`

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

1. **Overview**
   (The `...` operator, metavars, composition)

2. Semgrep In Practice
   (for antipatterns and business logic bugs)

3. Advanced Features
   (Extracting Routes, autofix, scripting)
Finding Banned or Deprecated Functions: **RC4** (... operator)

c, err := rc4.NewCipher(key)
d, err := rc4.NewCipher(otherKey)
e, err := rc4.NewCipher (    key    )

// rc4.NewCipher(key)
fmt.Println(“rc4.NewCipher(key)”)

⇒ https://semgrep.dev/s/10Bx

Full Solution: https://semgrep.live/X5g4 | docs
Finding Uses of `unsafe` (Metavarsiables)

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

⇒ [https://semgrep.dev/s/7gZe/](https://semgrep.dev/s/7gZe/)

Full Solution: [https://semgrep.dev/s/ErxF/](https://semgrep.dev/s/ErxF/)
Finding Insecure SSL Configurations  (Field/Param matching)

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

⇒ https://semgrep.live/Pewp

Full Solution: https://semgrep.live/4b9x
Finding Insecure SSL Configurations (Composing patterns)

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

⇒ https://semgrep.live/s/DbYd
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 `.yml` or `.yaml` 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`
  - `pattern-not`
  - `pattern-inside`
  - `pattern-not-inside`
  - `pattern-where-python`
- Metavariable Matching
  - Metavariables in Logical ANDs
  - Metavariables in Logical ORs
  - Metavariables in Complex Logic

https://github.com/returntocorp/semgrep/blob/develop/docs/configuration-files.md
Using Hardcoded Secret for JWT

```go
define jwtKey = []byte("my_secret_key")

token := jwt.NewWithClaims(jwt.SigningMethodHS256, claims)

tokenString, err := token.SignedString(jwtKey)
```
Demos

1. Overview
   (The `...` operator, metavars, composition)
2. **Semgrep In Practice**
   (for antipatterns and business logic bugs)
3. Advanced Features
   (Extracting Routes, autofix, scripting)
Hidden Goroutines (blog post) ⇒ https://semgrep.live/9A4z (Antipatterns)
Order of API Calls Must be Enforced (Business Logic)

```c
/*
 * 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](https://semgrep.live/6JqL)

Full Solution: [https://semgrep.live/oqZ6](https://semgrep.live/oqZ6)
Demos

1. Overview
   (The `...` operator, metavars, composition)

2. Semgrep In Practice
   (for antipatterns and business logic bugs)

3. **Advanced Features**
   (Extracting Routes, autofix, scripting)
func (a *App) initializeRoutes() {
    a.Router.HandleFunc("/products",
        a.getProducts).Methods("GET")
}

https://semgrep.dev/s/r6o1
Semgrep application: code inventory
Autofix - Insecure SSL Configuration

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

https://semgrep.dev/s/xxxA/
Scripting

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 a flag: --dangerously-allow-arbitrary-code-execution-from-rules
Use of Weak RSA Key

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

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

⇒ [https://semgrep.dev/s/zdRI](https://semgrep.dev/s/zdRI)

Full Solution: [https://semgrep.dev/s/zdRI](https://semgrep.dev/s/zdRI) | [docs](https://semgrep.dev/s/zdRI)
recap, a.k.a. "learn semgrep in 5 min"
#1 Code equivalence (**semantic grep**)

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Will match</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x == x$</td>
<td>(a+b != a+b) # &lt;=&gt; !(a+b==a+b)</td>
</tr>
<tr>
<td>foo(kwd1=1,kwd2=2,...)</td>
<td>foo(kwd2=2, kwd1=1, kwd3=3)</td>
</tr>
<tr>
<td>subprocess.open(...)</td>
<td>from subprocess import open as sub_open</td>
</tr>
<tr>
<td>import foo.bar</td>
<td>result = sub_open(&quot;ls&quot;)</td>
</tr>
<tr>
<td></td>
<td>from foo import bar</td>
</tr>
</tbody>
</table>

- **semgrep** knows about the semantics of the language, so one pattern can match variations of equivalent code (constant propagation! [https://semgrep.live/4K5](https://semgrep.live/4K5))
#2: ‘...’ ellipsis operator

- foo(...,5)  Will match  foo(1,2,3,4,5)  foo(5)
- foo(“...”)  Will match  foo(“whatever sequence of chars”)
- $V = get()
  ...
  eval($V)  Will match  user_data = get()
  print(“do stuff”)  foobar()
  eval(user_data)

‘...’ can match sequences of:

- Arguments, parameters
- Characters
- Statements
## Metavariables (part 1)

<table>
<thead>
<tr>
<th>Metavariable</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo($X,2)</td>
<td>Will match</td>
</tr>
<tr>
<td>if $E:</td>
<td>Will match</td>
</tr>
<tr>
<td>foo()</td>
<td></td>
</tr>
<tr>
<td>if $X &gt; $Y:</td>
<td>Will match</td>
</tr>
<tr>
<td>$S</td>
<td></td>
</tr>
<tr>
<td>$F(1,2)</td>
<td>Will match</td>
</tr>
</tbody>
</table>

- **Metavariables** start with a `$ ($X, $Y, $WHATEVER)`, contain uppercase ASCII characters
- **Matches:**
  - Expressions (including arguments)
  - Statements
  - Names (functions, fields, etc.)
You can reuse the same metavariable: **semgrep enforces equality constraint**
Awesome Use Cases

Search your code
- Vulnerabilities
- Audit security hotspots
- Extract routes
- Codify domain knowledge

Guard your code
- Secure defaults
- Banned APIs
- Best- and required-practices
- Configuration file auditing

Upgrade your code
- Migrate from deprecated APIs
- Apply automatic fixes
Search: Vulnerabilities
Scan lots of code
@$APP.route(...)

def $FUNC(..., $FILENAME, ...):
    ...
    open(<... $FILENAME ...>, ...)

https://semgrep.live/2Zz5/
Analyzer

dev/semgrep

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-talismann(0.0.1) 37
- toolk-flask-github(0.0.1) 1k

Step 3:
Select your parameters (optional)
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

<table>
<thead>
<tr>
<th>Repository</th>
<th>Commit</th>
<th>Findings</th>
<th>Annotations</th>
<th>Action</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 github.com/elantern/nodeunit</td>
<td>e8213b32</td>
<td>6</td>
<td></td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>2 github.com/rmmcmynn/banyan</td>
<td>5165cd03</td>
<td>6</td>
<td></td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>3 github.com/kangaroo-minifier</td>
<td>61389ab2</td>
<td>2</td>
<td></td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>4 github.com/alexkam/ramify</td>
<td>d23a56e2</td>
<td>14</td>
<td></td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>5 github.com/tivrin/nodemailer</td>
<td>9c7b6904</td>
<td>4</td>
<td></td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>6 github.com/innominate/nodemailer</td>
<td>0a5644c0</td>
<td>2</td>
<td></td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>7 github.com/innominate/nodemailer</td>
<td>7c6ba212</td>
<td>2</td>
<td></td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>8 github.com/innominate/nodemailer</td>
<td>ma203232</td>
<td>2</td>
<td></td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>9 github.com/innominate/nodemailer</td>
<td>213592b0</td>
<td>10</td>
<td></td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>10 github.com/innominate/nodemailer</td>
<td>0a16946b</td>
<td>2</td>
<td></td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>11 github.com/innominate/nodemailer</td>
<td>136a3b82</td>
<td>2</td>
<td></td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>12 github.com/innominate/nodemailer</td>
<td>8d6b262b</td>
<td>8</td>
<td></td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>13 github.com/innominate/nodemailer</td>
<td>f35458e0</td>
<td>4</td>
<td></td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>14 github.com/karma/karma</td>
<td>6c725e88</td>
<td>4</td>
<td></td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>15 github.com/karma/karma</td>
<td>7ba0d0a2</td>
<td>2</td>
<td></td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>16 github.com/karma/karma</td>
<td>931c61d2</td>
<td>4</td>
<td></td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>17 github.com/karma/karma</td>
<td>690a608b</td>
<td>2</td>
<td></td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>18 github.com/innominate/nodemailer</td>
<td>82a7b5c2</td>
<td>2</td>
<td></td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>19 github.com/innominate/nodemailer</td>
<td>4273c45d</td>
<td>2</td>
<td></td>
<td></td>
<td>results</td>
</tr>
</tbody>
</table>
@frontend.route('/api/update/get/<filename>', methods=['GET'])
def getZip(filename):
    return make_response(open(os.path.join(TEMPLATE_DIR, filename)).read())
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
Integrations - GitLab CI

```yaml
semgrep:
  image: returntocorp/semgrep-agent:v1
  script:
    - python -m semgrep_agent --config https://semgrep.dev/p/flask
```
Integrations - GitLab CI

- Linters
  - on: pull_request
- super-linter
- pre-commit
- semgrep with managed policy

Linters / semgrep with managed policy
failed 9 hour ago via GitLab CI

- ✔️ Set up job
- ✔️ Pull returntocorp/semgrep-action@v1
- ✔️ Run actions/checkout@v1
- ✗ Run returntocorp/semgrep-action@v1

```bash
GITUB_EVENT_NAME = $GITHUB_SERVER_URL = $GITHUB_API_URL = $GITHUB_GRAPHQL_URL
GITHUB_RUN_ID = $GITHUB_RUN_ID = $GITHUB_RUNTIME_URL = $GITHUB_RUNTIME_TOKEN = $GITHUB_CACHE_URL = $GITHUB_CACHE =
/home/runner/work/_temp/.github_home" "github/home" -v "home/runner/work/_temp/.github_home" "github/home" -v

# detected environment
- versions: semgrep v.17.0 on Python 3.8.5
- environment: running in github-actions, triggering event is 'pull_request'
- semgrep.dev: logged in as deployment #1

# setup 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_task.security.injection.path-traversal-open.path-traversal-open

- py: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
```
```python
python.flask.security.injection.path-traversal-open.path-traversal-open

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
```
registry
Community rule registry

community participation

- 700+ rules under development by r2c + community
- **NodeJsScan v4 is powered by semgrep!**
- [Gosec](https://github.com/Netflix/gosec) and [find-sec-bugs](https://findsecbugs.org) checks have been ported - no compilation required 👍
- Rule ideas contributed by Django co-creator
- Suggestions by Flask team
- Independent security researchers via HackerOne & elsewhere
semgrep rules by Damian Gryski, (author of Go-Perfbook)

github.com/dgryski/semgrep-go
Community rule registry

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

$ brew install semgrep
$ semgrep --config=<url>
$ semgrep --config=https://semgrep.dev/p/python
Coming Soon

1000 rules!

Semgrep Community!

Centrally manage Semgrep on your repos!

Tainting (intrafile)

\[\text{eval}($X:\text{<user\_data>})\]
Semgrep
lightweight static analysis for many languages

Locally:
1. (pip|brew) install semgrep
2. semgrep --config=r2c

Bence Nagy  |  bence@r2c.dev
r2c.dev  |  @r2cdev

https://r2c.dev/survey  ← plz :)

Online editor:
• semgrep.live