Enforcing Code & Security Standards with Semgrep

Colleen Dai | colleen@returntocorp.com

@r2cdev
So you’re doing a code review...

app/controllers/widget_controller.rb

```
85  +
86  +  before_action :ensure_user
```

Hey this looks great, but we've stopped using `ensure_user()`, could you use `ensure_logged_in()` instead?
So you’re doing a code review...

```ruby
app/controllers/widget_controller.rb

85 +
86 + before_action :ensure_user
```

You 3 days ago  Member

Hey this looks great, but we've stopped using `ensure_user()`, could you use `ensure_logged_in()` instead?

- Every code base has assumptions, requirements, coding standards
- Tools exist for language/framework generic checks
- What about code patterns **unique** to your project/org?
Semgrep tl;dr

- A customizable, lightweight, static analysis tool for finding bugs

Find bugs and enforce code standards.

Semgrep is a lightweight, offline, open-source, static analysis tool. Run community rules or write your own in less than 5 minutes. Configure and run in 2 minutes.
## Semgrep Trophy Case

<table>
<thead>
<tr>
<th>CVE</th>
<th>Semgrep rule</th>
<th>Affected software</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2019-5479</td>
<td>javascript.lang.security.detect-non-literal-require</td>
<td>larbitbase-api &lt; v0.5.5</td>
<td>An unintended require vulnerability in larbitbase-api may allow an attacker to load arbitrary non-production code (JavaScript file).</td>
</tr>
<tr>
<td>CVE-2020-8128</td>
<td>javascript.lang.security.detect-non-literal-require</td>
<td>jsreport &lt; 2.5.0</td>
<td>An unintended require and server-side request forgery vulnerabilities in jsreport version 2.5.0 and earlier allow attackers to execute arbitrary code.</td>
</tr>
<tr>
<td>CVE-2020-8129</td>
<td>javascript.lang.security.detect-non-literal-require</td>
<td>script-manager &lt; 0.8.6</td>
<td>An unintended require vulnerability in script-manager npm package version 0.8.6 and earlier may allow attackers to execute arbitrary code.</td>
</tr>
<tr>
<td>CVE-2020-7739</td>
<td>javascript.phantom.security.audit.phantom-injection</td>
<td>phantomjs-seo</td>
<td>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.</td>
</tr>
</tbody>
</table>
whois?

me:
Colleen Dai, security software engineer @ r2c
Graduated Stanford with B.S. of C.S., M.S. Stats

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. Semgrep Examples!

4. Integration into CI/CD

5. Semgrep Rules Registry
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
<table>
<thead>
<tr>
<th>Language</th>
<th>Status</th>
<th>Extensions</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Go</td>
<td>GA</td>
<td>.go</td>
<td>go, golang</td>
</tr>
<tr>
<td>Java</td>
<td>GA</td>
<td>.java</td>
<td>java</td>
</tr>
<tr>
<td>JavaScript</td>
<td>GA</td>
<td>.js, .jsx</td>
<td>js, jsx, javascript</td>
</tr>
<tr>
<td>JSON</td>
<td>GA</td>
<td>.json</td>
<td>json, JSON, Json</td>
</tr>
<tr>
<td>Python</td>
<td>GA</td>
<td>.py, .pyi</td>
<td>python, python2, python3, py</td>
</tr>
<tr>
<td>Ruby</td>
<td>GA</td>
<td>.rb</td>
<td>ruby, rb</td>
</tr>
<tr>
<td>TypeScript</td>
<td>GA</td>
<td>.ts, .tsx</td>
<td>ts, tsx, typescript</td>
</tr>
<tr>
<td>JSX</td>
<td>GA</td>
<td>.js, .jsx</td>
<td>js, jsx, javascript</td>
</tr>
<tr>
<td>TSX</td>
<td>GA</td>
<td>.ts, .tsx</td>
<td>ts, tsx, typescript</td>
</tr>
<tr>
<td>OCaml</td>
<td>alpha</td>
<td>.ml, .mli</td>
<td>ocaml, ml</td>
</tr>
<tr>
<td>PHP</td>
<td>alpha</td>
<td>.php</td>
<td>php</td>
</tr>
<tr>
<td>C</td>
<td>alpha</td>
<td>.c</td>
<td>c</td>
</tr>
<tr>
<td>Generic (YAML, ERB, Jinja, etc.)</td>
<td>alpha</td>
<td>*</td>
<td>generic</td>
</tr>
<tr>
<td>Rust</td>
<td>develop</td>
<td>.rs</td>
<td>rust, Rust, rs</td>
</tr>
<tr>
<td>Lua</td>
<td>develop</td>
<td>.lua</td>
<td>lua</td>
</tr>
</tbody>
</table>
Outline

1. Background
2. `grep` and Abstract Syntax Trees (ASTs)
3. Semgrep Examples!
4. Integration into CI/CD
5. Semgrep Rules Registry
grep, ASTs, and Semgrep

exec("ls")

exec(some_var)

exec  (arg)

exec(
    bar
)

other_exec(foo)

// exec(foo)

print("exec(bar)")
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

```
@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: 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 `eval()` in only 307 LOC 👍

https://github.com/eslint/eslint/blob/master/lib/rules/no-eval.js
Static Analysis at Scale: An Instagram Story

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. **Semgrep Examples!**
4. Integration into CI/CD
5. Semgrep Rules Registry
Tutorials

1. Ellipsis ("...") operator
2. Metavariables
3. Composing Patterns
4. Advanced Features
Finding Insecure Functions: Node Exec

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

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

Full Solution: [https://semgrep.live/1Kk](https://semgrep.live/1Kk)
Hard-coded Secrets, Constant String Arguments

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

⇒ https://semgrep.live/RGO8/

Full Solution: https://semgrep.live/A89w/
Tutorials

1. Ellipsis (“…”) operator
2. Metavariables
3. Composing Patterns
4. Advanced Features
Finding Uses of `unsafe` (Metavariables)

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

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

Full Solution: [https://semgrep.live/ZgLp](https://semgrep.live/ZgLp)
Path Traversal with send_file

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

⇒ [https://semgrep.live/4bXx](https://semgrep.live/4bXx)

Full Solution: [https://semgrep.live/Pevp](https://semgrep.live/Pevp)
Tutorials

1. Ellipsis ("...") operator
2. Metavariables
3. Composing Patterns 🎨
4. Advanced Features
@app.route("/index")
def index():
    r = response.set_cookie("username","drew")
    return r

⇒ https://semgrep.live/8dJ
Finding Insecure SSL Configurations

(Composing patterns)

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

⇒ [https://semgrep.dev/s/N4yN/](https://semgrep.dev/s/N4yN/)

Full Solution: [https://semgrep.live/Rewg](https://semgrep.live/Rewg)
Tutorials

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

[Full Solution: https://semgrep.live/oqZ6](https://semgrep.live/oqZ6)
Insecure SSL Configuration

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

https://semgrep.dev/xxxA/
func bad1() {
}

pattern: |
    http.NewRequest(..., "=~/[hH][tT][tT][pP]://.*/", ...)
func (a *App) initializeRoutes() {
    a.Router.HandleFunc("/products",
        a.getProduct).Methods("GET")
}
Semgrep application: code inventory
resource "aws_s3_bucket" "b" {
  bucket = "my-tf-test-bucket"
  acl    = "public-read-write"
...
}

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

https://semgrep.dev/s/neOZ/
Pattern syntax

This document describes Semgrep's pattern syntax. You can also see pattern examples by language. In the command line, patterns are specified with the flag `--pattern` (or `--e`). Multiple coordinating patterns may be specified in a configuration file. See rule syntax for more information.

- Expression matching
- String matching
- Ellipsis operator
  - Function calls
  - Method calls
  - Function definitions
  - Class definitions
  - Strings
  - Binary operations
  - Arrays
  - Conditionals and loops
- Metavariabes
- Typed Metavariabes
- Equivalences
  - Imports
  - Constants
- Deep expression operator
- Limitations
  - Statements types
  - Partial statements

https://semgrep.dev/docs/writing-rules/pattern-syntax/
Search: Vulnerabilities
@$APP.route(...)

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

https://semgrep.live/2Zz5/
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>
</tr>
</thead>
<tbody>
<tr>
<td>[GitHub URL]</td>
<td>cd773a2</td>
<td>2</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>ee713d8</td>
<td>6</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>51e11f1</td>
<td>6</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>a135bbd</td>
<td>2</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>429e662</td>
<td>2</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>9f354f44</td>
<td>6</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>20e0a5a2</td>
<td>2</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>65b370c</td>
<td>8</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>6e44f45a</td>
<td>4</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>6235ed8</td>
<td>4</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>70a5b04</td>
<td>2</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>9e106d2</td>
<td>4</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>6497b6d</td>
<td>2</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>dd4bb00</td>
<td>2</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>ba5f1a1d</td>
<td>2</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>a69231d</td>
<td>2</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>67a53ee</td>
<td>2</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>d4e129c</td>
<td>289</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>b94f1f63</td>
<td>2</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>2199819</td>
<td>2</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>b05f5d</td>
<td>2</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>ca9b302</td>
<td>14</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>2d6c6eb</td>
<td>4</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>813b988</td>
<td>4</td>
<td></td>
<td>results</td>
</tr>
<tr>
<td>[GitHub URL]</td>
<td>4a3b338</td>
<td>12</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())
Outline

1. Background
2. *grep* and Abstract Syntax Trees (ASTs)
3. Semgrep Examples!
4. *Integration into CI/CD*
5. Semgrep Rules Registry
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

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

Linters / semgrep with managed policy
failed 1 hour ago in 1m 26s

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

```bash
GITHUB_EVENT_NAME -> GITHUB_SERVER_URL -> GITHUB_API_URL -> GITHUB_GRAPHQL_URL
-> ACTIONS_RUNTIME_URL -> ACTIONS_RUNTIME_TOKEN -> ACTIONS_CACHE_URL -> GITHUB_
"/home/runner/work/_temp/semgrep_home":/github/home" -> v"/home/runner/work/_to
returntocorp/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 |   | open(path).readlines(), mimetype="text/plain"
23 |   | Found request data in a call to 'open'. Ensure the request data is validated or sanitized, otherwise it could result in path traversal attacks.
24 |   | exiting with failing status
25 |   | Complete job
```
Outline

1. Background
2. `grep` and Abstract Syntax Trees (ASTs)
3. Semgrep Examples!
4. Integration into CI/CD
5. Semgrep Rules Registry
Community rule registry

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

### findsecbugs
by r2c

Selected rules from FindSecBugs, a security checker for Java, rewritten in Semgrep.

- Java

---

### gosec
by Uizii Otgonbaatar

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

- Go

---

### dgryski.semgrep-go
by Damian Gryski

Rules for finding odd Go code. See [github.com/dgryski/semgrep-go](https://github.com/dgryski/semgrep-go) to contribute.

- Go

---

### nodejsscan
by Ajin Abraham

Rules from the preeminent Node.js security scanner, NodeJSScan.

- JavaScript
## Languages and Frameworks

Get security coverage for the languages and frameworks you use.

<table>
<thead>
<tr>
<th>Framework</th>
<th>Default ruleset</th>
</tr>
</thead>
<tbody>
<tr>
<td>minusworld.ruby-all</td>
<td>Default ruleset for Ruby, by r2c</td>
</tr>
<tr>
<td>python</td>
<td>Default ruleset for Python, by r2c</td>
</tr>
<tr>
<td>javascript</td>
<td>Default ruleset for JavaScript, by r2c</td>
</tr>
</tbody>
</table>

### r2c
- Go
- Java
- JavaScript
- Python

Default ruleset, by r2c
- audit
- crypto
- cookies
- correctness
- injection
- security
- spring
- xss
- xxe

### r2c-ci
- Go
- Java
- JavaScript
- Python

Scan for runtime errors, logic bus, and high-confidence security vulnerabilities.
- CI
- cookies
- correctness
- crypto
- csrf
- injection
- security
- spring
- xss
- xxe
- logic
- logic bugs

### r2c-security-audit
- Ruby
- JavaScript
- Go
- Java
- C

Scan code for potential security issues that require additional review. Recommended for team.
- security
- audit
- xxe
- injection
deserialization
- xss
- jwt
- csrf
- crypto

$ semgrep --config=https://semgrep.dev/p/r2c
Partnering with OWASP

- New partnership between Semgrep + OWASP ASVS, Cheat Sheets
- **Goal**: Out of the box support for:
  - Verifying if your code is compliant with ASVS Level 1
  - Finding code that violates Cheat Sheets best practice recommendations

Want to get involved? Let’s talk! 🙌

Thanks to Daniel Cuthbert, Joe Bollen, Rohit Salecha, and more

[GitHub Repository: semgrep/rules-owasp-asvs]

Update: Adding Semgrep Rules #457
Coming Soon

Improved language support, new languages

More rules! + prevention cheatsheets

Semgrep Community!

Centrally manage Semgrep on your repos!

VS Code extension (in beta)!

Tainting + constant propagation + speedups!
Wrapping Up
Semgrep
lightweight static analysis for many languages

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

Playground:
• semgrep.live
Questions?

Colleen Dai  | colleen@returntocorp.com
r2c.dev     | @r2cdev

https://r2c.dev/survey ← plz :)

Semgrep
lightweight static analysis for many languages