introducing Semgrep

Isaac Evans | r2c.dev

@r2cdev

slides will be posted at semgrep.dev
tl;dw – This Talk

- Writing secure code is hard
- SAST tools are too noisy / too slow, grep isn’t expressive enough
- 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:
Isaac Evans, co-founder @ r2c
formerly: BS/MS MIT, MIT Lincoln Labs, US DoD

r2c
We’re a static analysis startup based in SF and we care about software security.
Outline

1. Writing secure code is hard
2. Getting started with Semgrep
3. Tutorials
4. Using Semgrep on your code & CI
It’s really hard to write secure code

```
set_cookie(key, value='', max_age=None, expires=None, path='/', domain=None, secure=False, httponly=False, samesite=None)
```

Sets a cookie. The parameters are the same as in the cookie `Morsel` object in the Python standard library but it accepts unicode data, too.

A warning is raised if the size of the cookie header exceeds `max_cookie_size`, but the header will still 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,
@app.route("/index")
def index():
    resp = response.set_cookie("username","isaace")
    return resp
How might we detect this? ⇒ REGEX

```bash
grep -RE 'response\.set_cookie\(' path/to/code
```
response.set_cookie("username","isaace")

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

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

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

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

areponse.set_cookie(cleared_cookie)
response.set_cookie("username","DrewDennison")

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

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

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

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

1 True Positive

4 False Positives

1 False Negative
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
```
Linters & SAST: Solution?

- NodeJSScan, dlint, Bandit, Flake8, eslint, tslint, pylint, Golint, gosec, rubocop, among others...
- Have to become an **expert in every AST syntax** for every language my team uses
- Need **programming language expertise** to cover all idioms: languages have “more than one way to do it”
- **Commercial SAST tools?**
  - Expensive
  - Hard to tune
flask.response.set_cookie(<whatever>)
Complexity

grep  Semgrep  SAST
Semgrep – the power of SAST with the ease of grep

$ brew install returntocorp/semgrep/semgrep
$ semgrep -lang python -e 'flask.request.set_cookie(...)' /path/to/my/project

⇒ free & open source

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

Yoann Padialleau, original Semgrep author and first program analysis hire at Facebook, joined r2c last year. Previously PhD @ Inria, contributor to coccinelle.lip6.fr
Getting Started

**semgrep.dev** ⇒ github.com/returntocorp/semgrep
Tutorials
Tutorial: Node Exec

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

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

Full Solution: [https://semgrep.live/1Kk](https://semgrep.live/1Kk)
func main() {
    http.HandleFunc("/index", Handler)
    http.ListenAndServe(":80", nil)
}

⇒ https://semgrep.live/dyP

Full Solution: https://semgrep.live/Zy7
Tutorial: Always True

```python
if foo == foo:
    print("foobar")
else:
    print("baz")
```

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

Tutorial: Injection Sending 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/7dv](https://semgrep.live/7dv)

Full Solution: [https://semgrep.live/EN8](https://semgrep.live/EN8)
Tutorial: 🍪 problem

```python
@app.route("/index")
def index():
    r = response.set_cookie("username","isaace")
    return r
```

⇒ https://semgrep.live/8dJ
Tutorial: 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/LNX

Full Solution: https://semgrep.live/kle
Tutorial: exit()

```python
if username == "ievans":
    print("n00b")
    exit(1)
else:
    print("l33t")
```

⇒ https://semgrep.live/z3P

Full Solution: https://semgrep.live/ppJ
recap, a.k.a. "learn semgrep in 5 min"
#1 Code equivalence (semantically grep)

- `$X == $X` will match `(a+b != a+b) # <=> !(a+b==a+b)`
- `foo(kwd1=1,kwd2=2,...)` will match `foo(kwd2=2, kwd1=1, kwd3=3)`
- `subprocess.open(...)` will match `from subprocess import open as sub_open
result = sub_open("ls")`
- `import foo.bar` will match `from foo import bar`

- **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("...")` will match
- `$V = get()
  ...
  eval($V)` will match

‘...’ can match sequences of:

- Arguments, parameters
- Characters
- Statements
#3 Metavariabes (part 1)

- **Metavariabes** start with a $ ($X, $Y, $WHATEVER), contain uppercase ASCII characters

- **Matches:**
  - Expressions (including arguments)
  - Statements
  - Names (functions, fields, etc.)
#3 Metavarniables (Part 2)

You can reuse the same metavariable: `semgrep` enforces `equality constraint`
## Language Support

<table>
<thead>
<tr>
<th>Language</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>✓</td>
</tr>
<tr>
<td>Javascript</td>
<td>✓</td>
</tr>
<tr>
<td>Go</td>
<td>✓</td>
</tr>
<tr>
<td>Java</td>
<td>✓</td>
</tr>
<tr>
<td>C</td>
<td>✓</td>
</tr>
<tr>
<td>Typescript</td>
<td>Coming...</td>
</tr>
<tr>
<td>PHP</td>
<td>Coming...</td>
</tr>
</tbody>
</table>

Branch: `develop` / `semgrep / LICENSE`

returntocorp/semgrep is licensed under the 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.
registry beta
Community rule registry

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

community participation

- 100s of rules under development by r2c + community
- NodeJsScan v4 is powered by semgrep!
- Rule ideas contributed by Django co-creator
- Suggestions by Flask team
- Independent security researchers via HackerOne & elsewhere

```bash
$ brew install returntocorp/semgrep/semgrep
$ semgrep --config=<folder of yaml files>
```
Community rule registry

community-contributed coverage for OWASP top 10 across most languages!
tuning rules
iterating on a rule: v1

/your/project/.sgrep.yml

rules:
- id: stupid_equal
  pattern: $X == $X
  message: $X == $X is always true
  languages: [python]
  severity: WARNING

15
16 def handler(level, func, path, line, msg):
17     assert 1 == 1
18
iterating on a rule: v2

/your/project/.sgrep.yml

rules:
- id: stupid_equal
  patterns:
    pattern-not-inside: assert(...) 
    pattern: $X == $X
    message: $X == $X is always true
  languages: [python]
  severity: WARNING
reducing false positives at scale

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] github.com/secdev/scapy/blob/8066a9d87165b7c5387cc13e141f58cc2603dc1a/scapy/contrib/isotp.py#L364

[2] github.com/apache/libcloud/blob/6540c127199f92465c4917a974f1fbd06bc71d27/libcloud/compute/d rivers/cloudsigma.py#L612

[3] github.com/Azure/azure-sdk-for-python/blob/76835cddacb9e6996dace0ee686ae5a7c446c5f8/sdk/c osmos/azure-cosmos/azure/cosmos/routing/routin g_map_provider.py#L97
iterating on a rule: other languages

/your/project/.sgrep.yml

rules:
  - id: useless-assignment
    pattern: |
        $X = $Y;
        $X = $Z;
    message: "$X` is assigned twice; useless first assignment"
    languages: [js]
    severity: WARNING
iterating on a rule: scriptable!

```
.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 non-default flag: --dangerously-allow-arbitrary-code-execution-from-rules
Deeply configurable and scriptable

Semgrep.dev has documentation on the full configuration syntax.

Semgrep is available as a Python library (pip-installable!)

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
- Optional Fields
  - fix
  - metadata
  - paths
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

CircleCI Config

```yaml
jobs:
  build:
    docker:
      - image: returntocorp/semgrep:develop
    working_directory: /home/repo
    steps:
      - checkout
      - run: semgrep --config=https://sgrep.live/c/r/r2c.python.django
```

Github Action

```
semgrep action
This action runs semgrep and returns the output

Inputs

<table>
<thead>
<tr>
<th>config</th>
</tr>
</thead>
<tbody>
<tr>
<td>The config file</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>The output arg file</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>The target(s) to scan</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>error</th>
</tr>
</thead>
<tbody>
<tr>
<td>If true will exit 1, which will break the build.</td>
</tr>
</tbody>
</table>
```
Semgrep Core Features Coming Soon

More code equivalences!

$X + Y \iff Y + X$

Integrate typing information

new Buffer($X$:int, …)

($O$:XmlHttpRequest).get(…)

Support for $LANG$

Taking votes for next language
Help us make Semgrep better!

We have many rules for specific frameworks for OWASP Top 10 issues in the registry! Take our survey about the talk and we’ll email you a curated selection:

r2c.dev/survey

use
r2c is building a next-gen hosted SAST product with Semgrep at the core: talk to us if you're not in love with your SAST tool!

contribute
Super easy to share rules — contribute your security expertise →
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, C

Run it now!
1. Semgrep.live
2. brew install returntocorp/semgrep/semgrep
3. docker run --rm -v "${PWD}:/home/repo" returntocorp/semgrep

Isaac Evans | isaac@r2c.dev
r2c.dev | @r2cdev

https://r2c.dev/survey