semGrep 101

Nishaanth G

Agenda

- What is Semgrep?
- Operators in Semgrep
- Conditional Operations
- Examples of existing rules
- Conclusion

What is Semgrep?

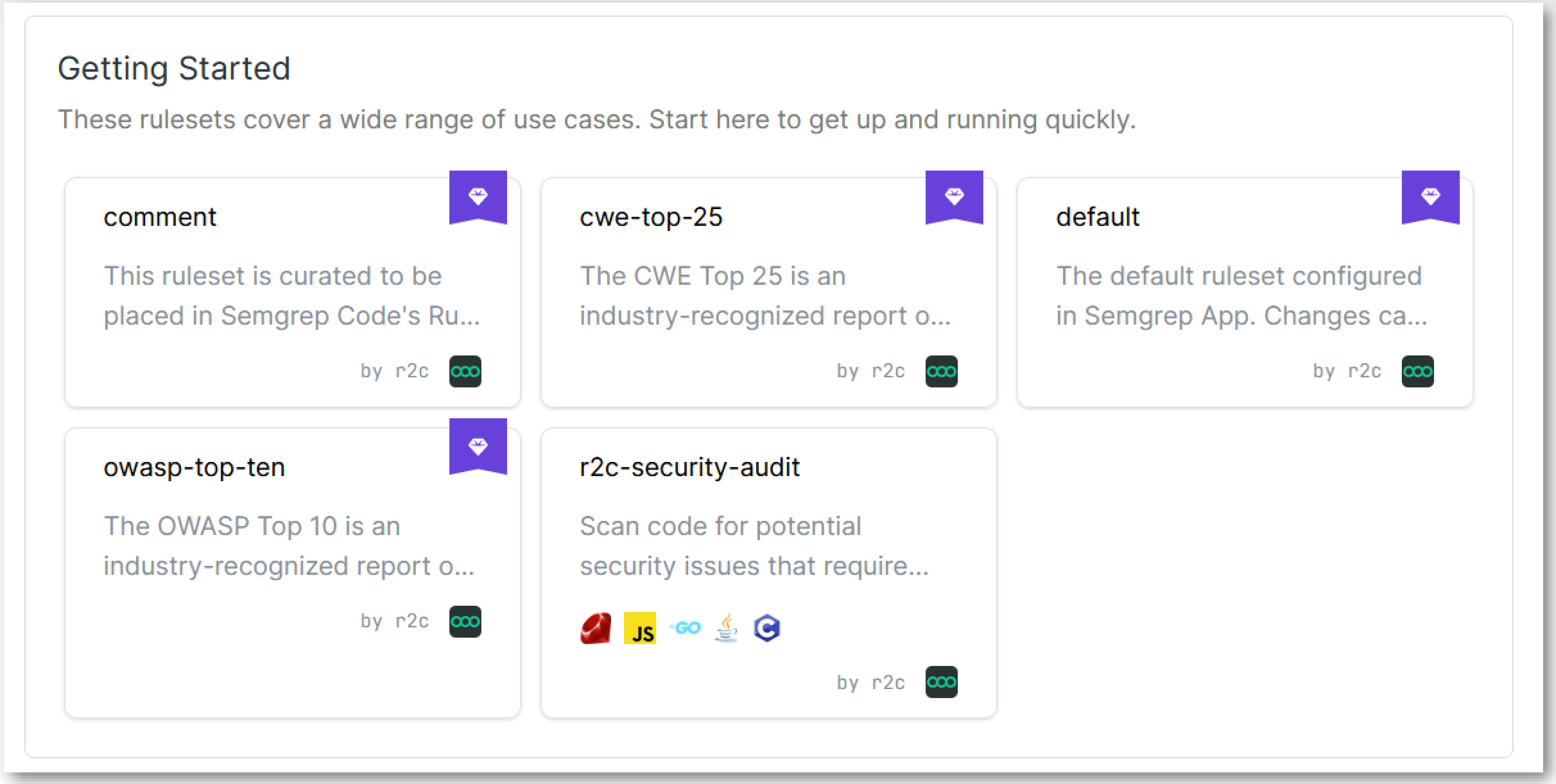
- Semgrep is fast, open-source static analysis engine for finding bugs and enforcing code standards across the codebase. Code is analysed locally and not uploaded to cloud.
- Simple syntax, very similar to writing a code snippet. No need of complex regular expressions, abstract syntax trees or the have access to have buildable code (like CodeQL?)
- Semgrep registry contains over 2000+ rules for various languages (has support for over 30+ languages) and vulnerabilities.

simple mode
advanced mode
test code
metadata
docs
Pro Engine beta

```yaml
rules:
  - id: print-session-data
    pattern: print(session.data)
    message: Use logging.debug() instead of print()
    languages:
      - python
    severity: INFO
    fix: logger.debug($...X)

```

```python
def start_server():
    logger.info('starting skynet')
    skynet.init()
    # TODO Change this to logging framework before prod
    print(f'--> debug, skynet init vector is {skynet.iv}')
    print(session.data)
    return skynet.rule_forever()
```

What is Semgrep?



Getting Started

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

comment

This ruleset is curated to be placed in Semgrep Code's Ru...

by r2c

cwe-top-25

The CWE Top 25 is an industry-recognized report o...

by r2c

default

The default ruleset configured in Semgrep App. Changes ca...

by r2c

owasp-top-ten

The OWASP Top 10 is an industry-recognized report o...

by r2c

r2c-security-audit

Scan code for potential security issues that require...

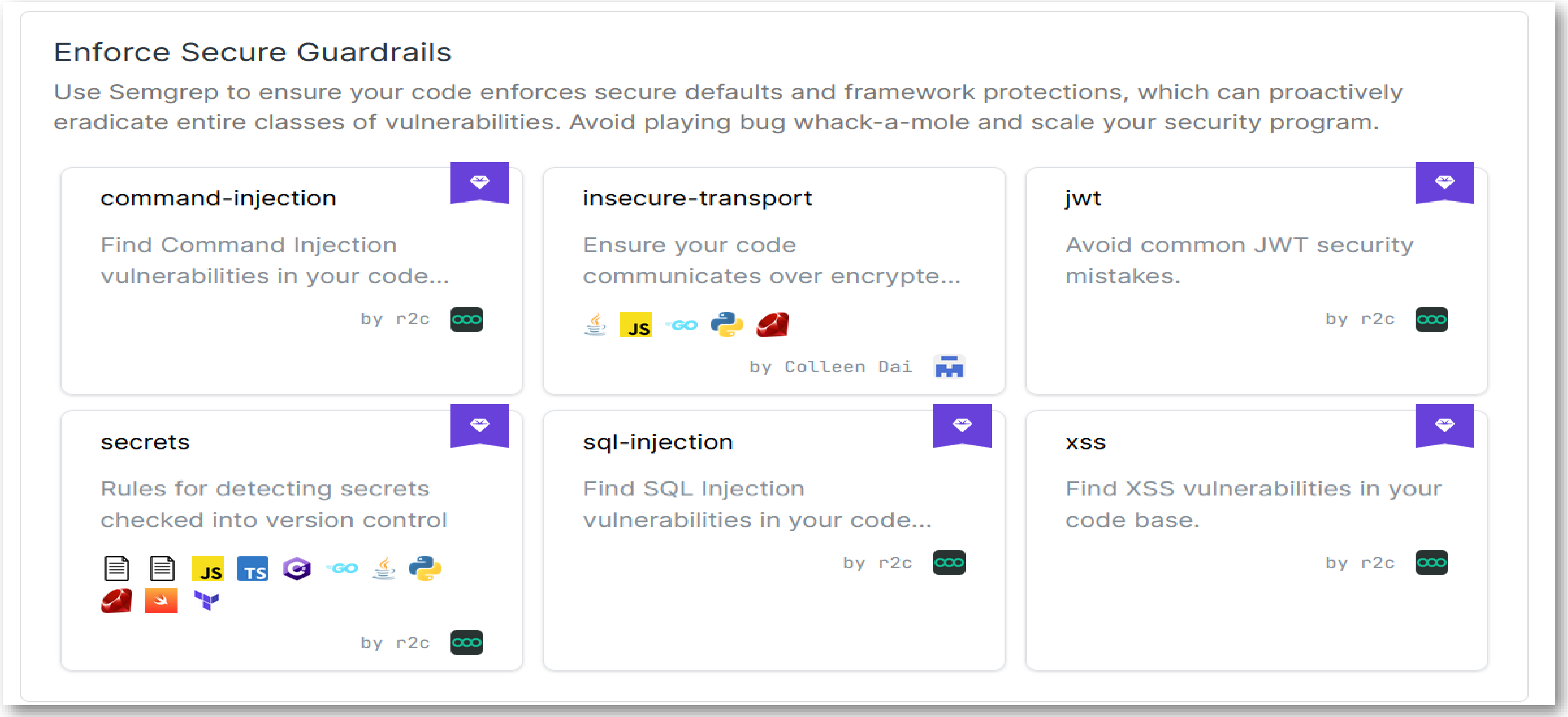
by r2c

semgrep --config "p/cwe-top-25"

What is Semgrep?

MDSec

$ semgrep --config “p/command-injection”



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.

command-injection
Find Command Injection vulnerabilities in your code...
by r2c

insecure-transport
Ensure your code communicates over encrypte...
by Colleen Dai

jwt
Avoid common JWT security mistakes.
by r2c

secrets
Rules for detecting secrets checked into version control
by r2c

sql-injection
Find SQL Injection vulnerabilities in your code...
by r2c

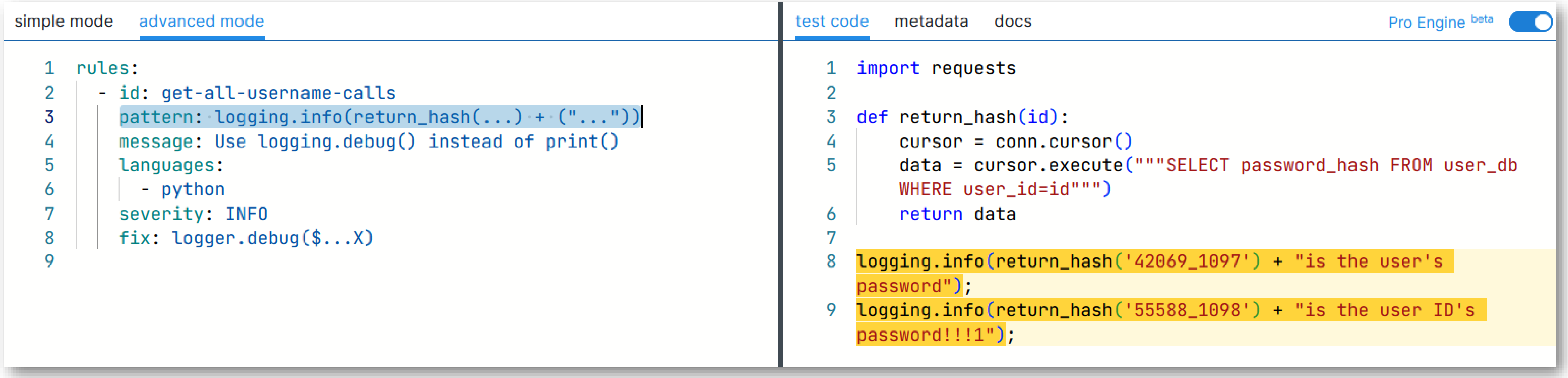
xss
Find XSS vulnerabilities in your code base.
by r2c

kubectl run nginx --image=nginx:1.15.8 --replicas=2 --port=80
kubectl run nginx --image=nginx:1.15.8 --replicas=2 --port=80
kubectl run nginx --image=nginx:1.15.8 --replicas=2 --port=80
kubectl run nginx --image=nginx:1.15.8 --replicas=2 --port=80
kubectl run nginx --image=nginx:1.15.8 --replicas=2 --port=80

Operators in Semgrep

MDSec

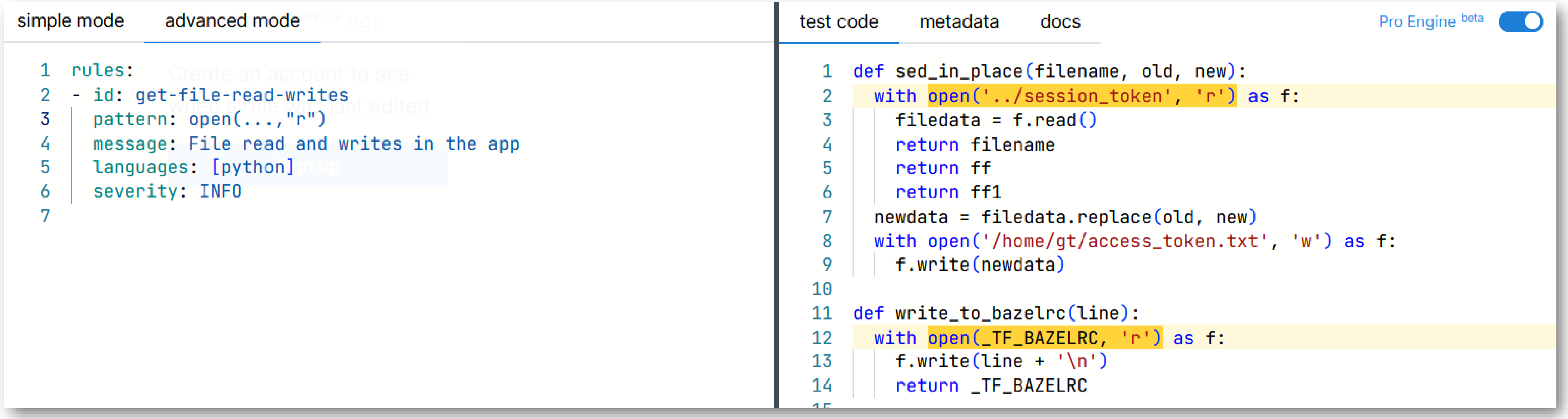
- Ellipsis operator (...) is an abstraction which allows you to skip over characters (variables, arguments) which we do not care about (similar to a .* regular expression)
- An example use-case would be to find all the instances of a overloaded function/method with different type of values getting passed to the function.
- In case you want to match any constant string, use ellipsis inside quotes ("...")



```yaml
rules:
  - id: get-all-username-calls
    pattern: logging.info(return_hash(...) + ("..."))
    message: Use logging.debug() instead of print()
    languages:
      - python
    severity: INFO
    fix: logger.debug($...X)
```

```python
import requests

def return_hash(id):
    cursor = conn.cursor()
    data = cursor.execute("""SELECT password_hash FROM user_db
    WHERE user_id=id""")
    return data

logging.info(return_hash('42069_1097') + "is the user's
password");
logging.info(return_hash('55588_1098') + "is the user ID's
password!!!1");
```

Ellipsis operator can also be used in between code to match whatever is between the block. The position/order of the value matters when you search. For instance, the pattern *open("r")* will not match the code *open("some_filename.csv", "r")* since the order the values don't match. Ellipsis can be used on both the sides of the pattern, something like *open(…,"r",…)* to match all file reads.



```yaml
rules:
- id: get-file-read-writes
  pattern: open(...,"r")
  message: File read and writes in the app
  languages: [python]
  severity: INFO
```

```python
def sed_in_place(filename, old, new):
  with open('../session_token', 'r') as f:
    filedata = f.read()
    return filename
    return ff
    return ff1
  newdata = filedata.replace(old, new)
  with open('/home/gt/access_token.txt', 'w') as f:
    f.write(newdata)

def write_to_bazelrc(line):
  with open(_TF_BAZELRC, 'r') as f:
    f.write(line + '\n')
    return _TF_BAZELRC
```

Operators in Semgrep

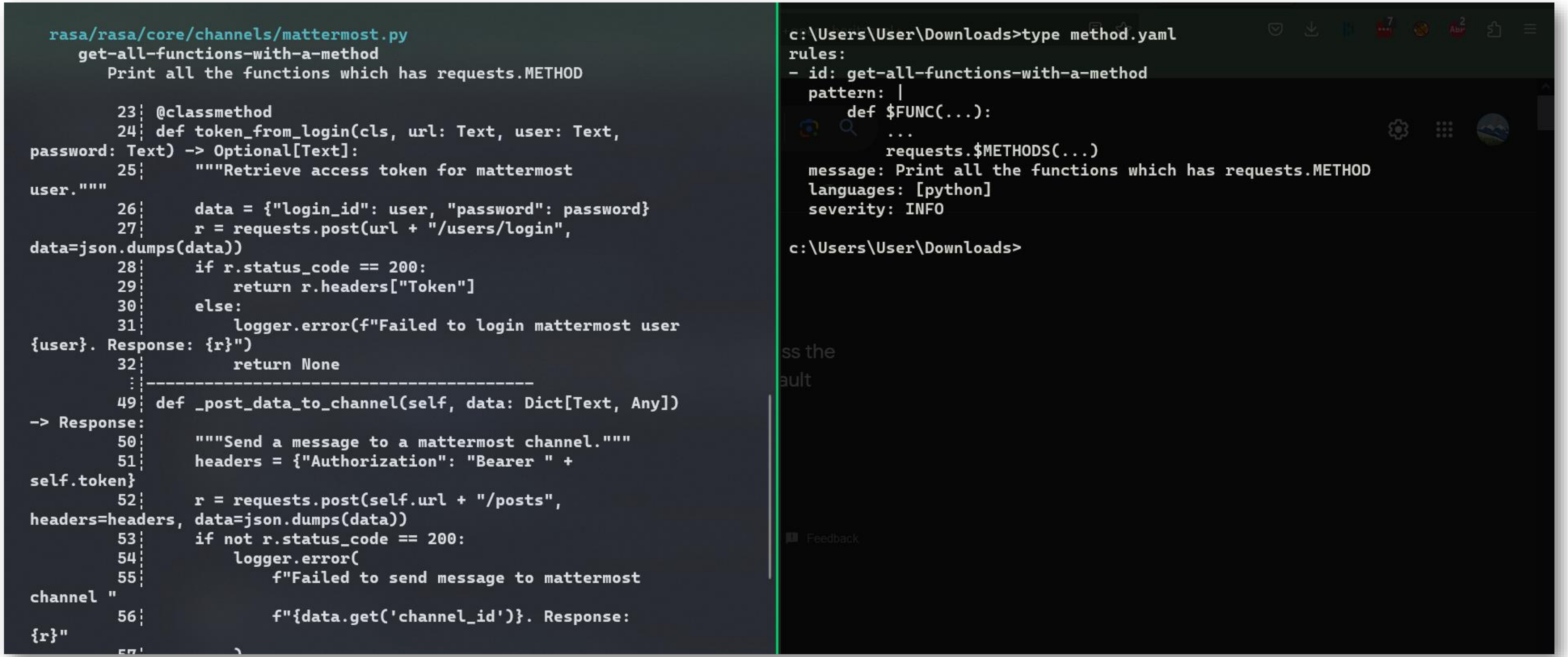
simple mode
advanced mode
test code
metadata
docs
Pro Engine beta

```yaml
rules:
- id: get-code-between-parser
  pattern: |
      etree.XMLParser()
      ...
      return jsonify(processed_xml)
  message: XML check if anything is written
  languages: [python]
  severity: INFO
```

```python
@tools.route("/is_xml", methods=['POST'])
def tools_is_xml():
    try:
        xml_raw = request.files['xml'].read()
        parser = etree.XMLParser()
        root = etree.fromstring(xml_raw, parser)
        xml = etree.tostring(root, pretty_print=True,
        encoding='unicode')
        return jsonify(processed_xml)
    except Exception as e:
        return jsonify({'status': 'no', 'message': str(e)})
```

Operators in Semgrep

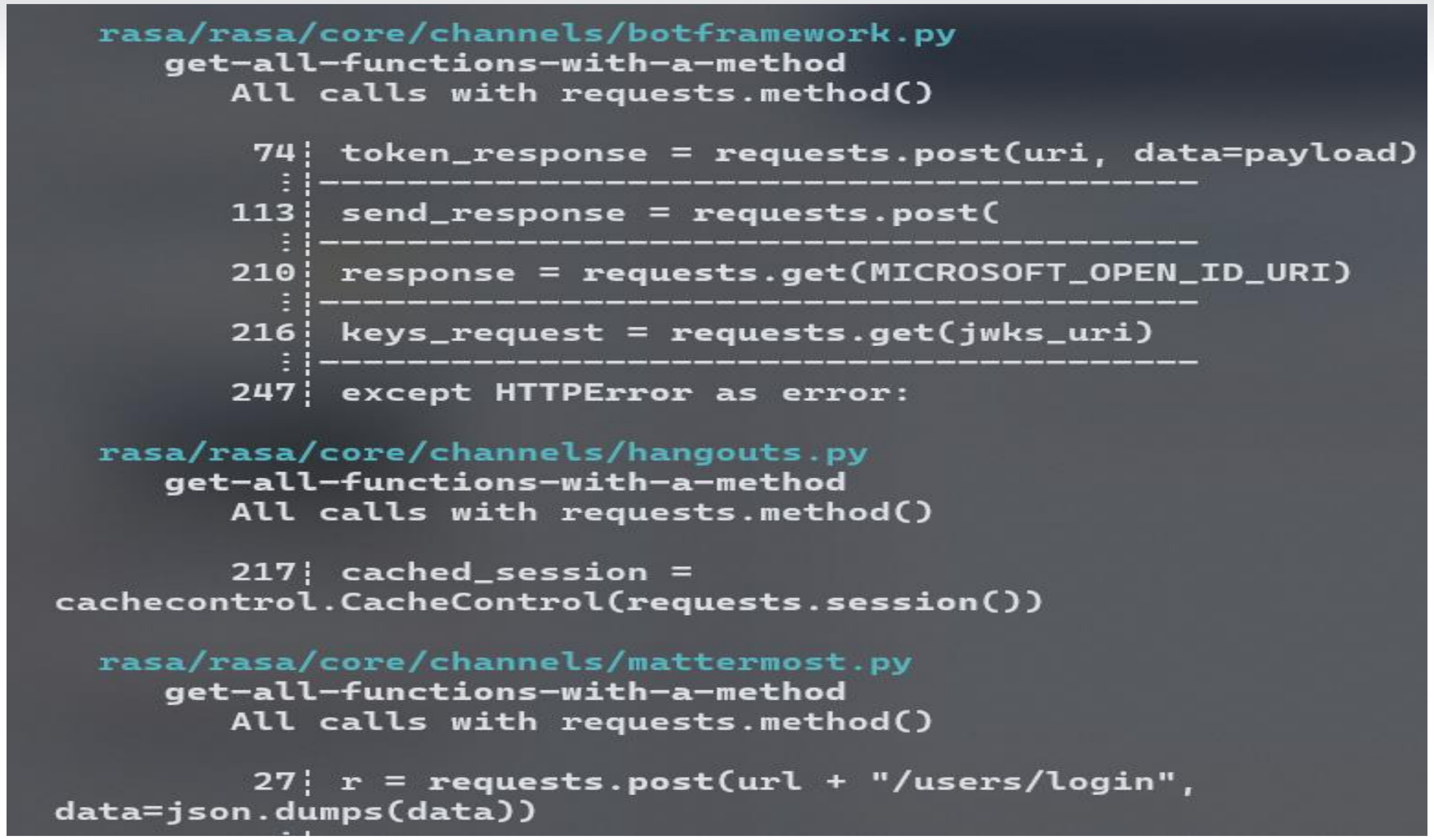
Metavariables, similar to regular expression capture groups are used when we don't know what exactly we want to match (like a function name which we don't know in advance). Always starts with a dollar sign and can contain uppercase characters, digits and underscores.



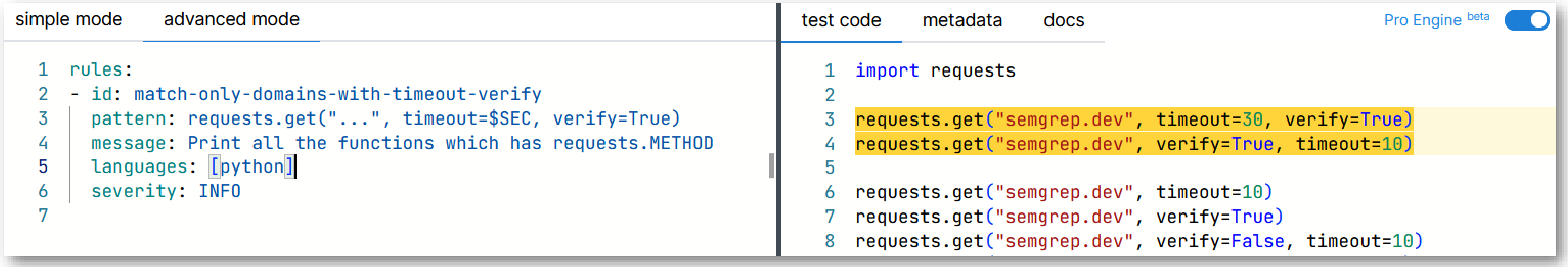
```
  rasa/rasa/core/channels/mattermost.py
     get-all-functions-with-a-method
        Print all the functions which has requests.METHOD

          @classmethod
          def token_from_login(cls, url: Text, user: Text,
password: Text) -> Optional[Text]:
              """Retrieve access token for mattermost
user."""
              data = {"login_id": user, "password": password}
              r = requests.post(url + "/users/login",
data=json.dumps(data))
              if r.status_code == 200:
                  return r.headers["Token"]
              else:
                  logger.error(f"Failed to login mattermost user
{user}. Response: {r}")
                  return None
         ⋮----------------------------------------
          def _post_data_to_channel(self, data: Dict[Text, Any])
-> Response:
              """Send a message to a mattermost channel."""
              headers = {"Authorization": "Bearer " +
self.token}
              r = requests.post(self.url + "/posts",
headers=headers, data=json.dumps(data))
              if not r.status_code == 200:
                  logger.error(
                      f"Failed to send message to mattermost
channel "
                      f"{data.get('channel_id')}. Response:
{r}"
```

```
c:\Users\User\Downloads>type method.yaml
rules:
- id: get-all-functions-with-a-method
  pattern: |
      def $FUNC(...):
        ...
        requests.$METHODS(...)
  message: Print all the functions which has requests.METHOD
  languages: [python]
  severity: INFO

c:\Users\User\Downloads>
```

Operators in Semgrep



```
  rasa/rasa/core/channels/botframework.py
     get-all-functions-with-a-method
        All calls with requests.method()

          74┆ token_response = requests.post(uri, data=payload)
           ⋮┆----------------------------------------
         113┆ send_response = requests.post(
           ⋮┆----------------------------------------
         210┆ response = requests.get(MICROSOFT_OPEN_ID_URI)
           ⋮┆----------------------------------------
         216┆ keys_request = requests.get(jwks_uri)
           ⋮┆----------------------------------------
         247┆ except HTTPError as error:

  rasa/rasa/core/channels/hangouts.py
     get-all-functions-with-a-method
        All calls with requests.method()

         217┆ cached_session =
cachecontrol.CacheControl(requests.session())

  rasa/rasa/core/channels/mattermost.py
     get-all-functions-with-a-method
        All calls with requests.method()

          27┆ r = requests.post(url + "/users/login",
data=json.dumps(data))
```

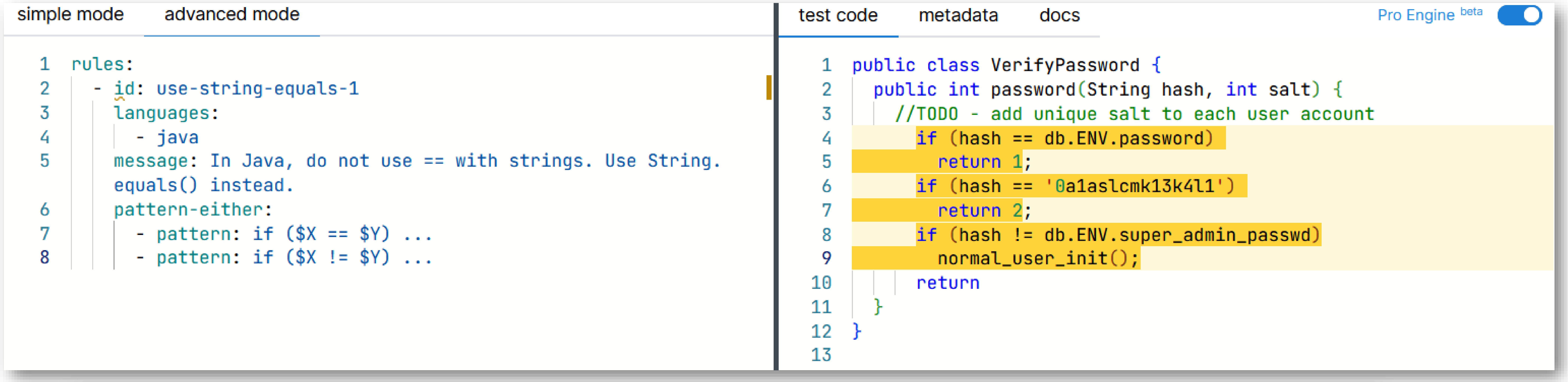


```yaml
rules:
- id: match-only-domains-with-timeout-verify
  pattern: requests.get("...", timeout=$SEC, verify=True)
  message: Print all the functions which has requests.METHOD
  languages: [python]
  severity: INFO
```

```python
import requests

requests.get("semgrep.dev", timeout=30, verify=True)
requests.get("semgrep.dev", verify=True, timeout=10)

requests.get("semgrep.dev", timeout=10)
requests.get("semgrep.dev", verify=True)
requests.get("semgrep.dev", verify=False, timeout=10)
```

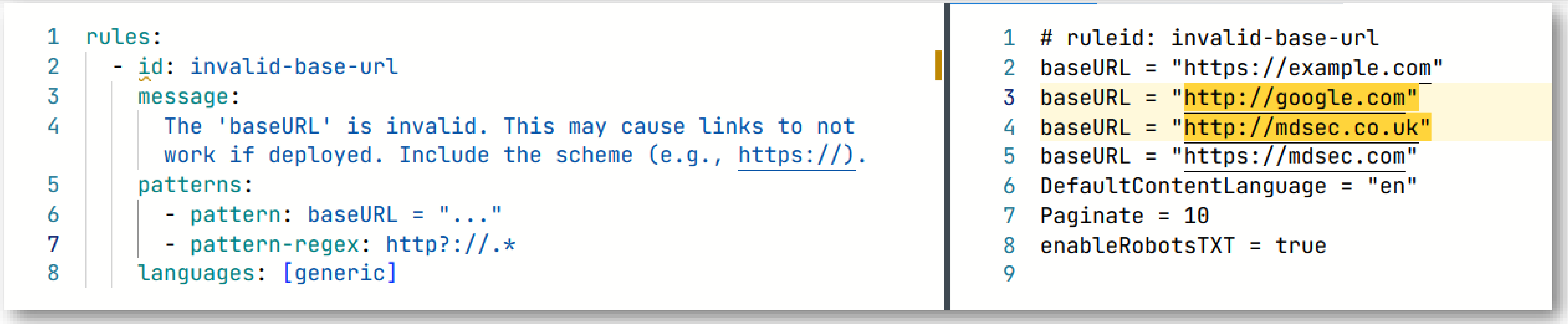
Conditional Operators in Semgrep

Semgrep has support for logical operators as other programming languages. Possible values include pattern-either, pattern-not, pattern-inside, pattern-not-inside, metavariable-regex.



```
rules:
  - id: use-string-equals-1
    languages:
      - java
    message: In Java, do not use == with strings. Use String.
    equals() instead.
    pattern-either:
      - pattern: if ($X == $Y) ...
      - pattern: if ($X != $Y) ...
```

```
public class VerifyPassword {
  public int password(String hash, int salt) {
    //TODO - add unique salt to each user account
      if (hash == db.ENV.password)
        return 1;
      if (hash == '0a1aslcmk13k4l1')
        return 2;
      if (hash != db.ENV.super_admin_passwd)
        normal_user_init();
      return
  }
}
```



```
rules:
  - id: invalid-base-url
    message:
      The 'baseURL' is invalid. This may cause links to not
      work if deployed. Include the scheme (e.g., https://).
    patterns:
      - pattern: baseURL = "..."
      - pattern-regex: http?://.*
    languages: [generic]
```

```
# ruleid: invalid-base-url
baseURL = "https://example.com"
baseURL = "http://google.com"
baseURL = "http://mdsec.co.uk"
baseURL = "https://mdsec.com"
DefaultContentLanguage = "en"
Paginate = 10
enableRobotsTXT = true
```

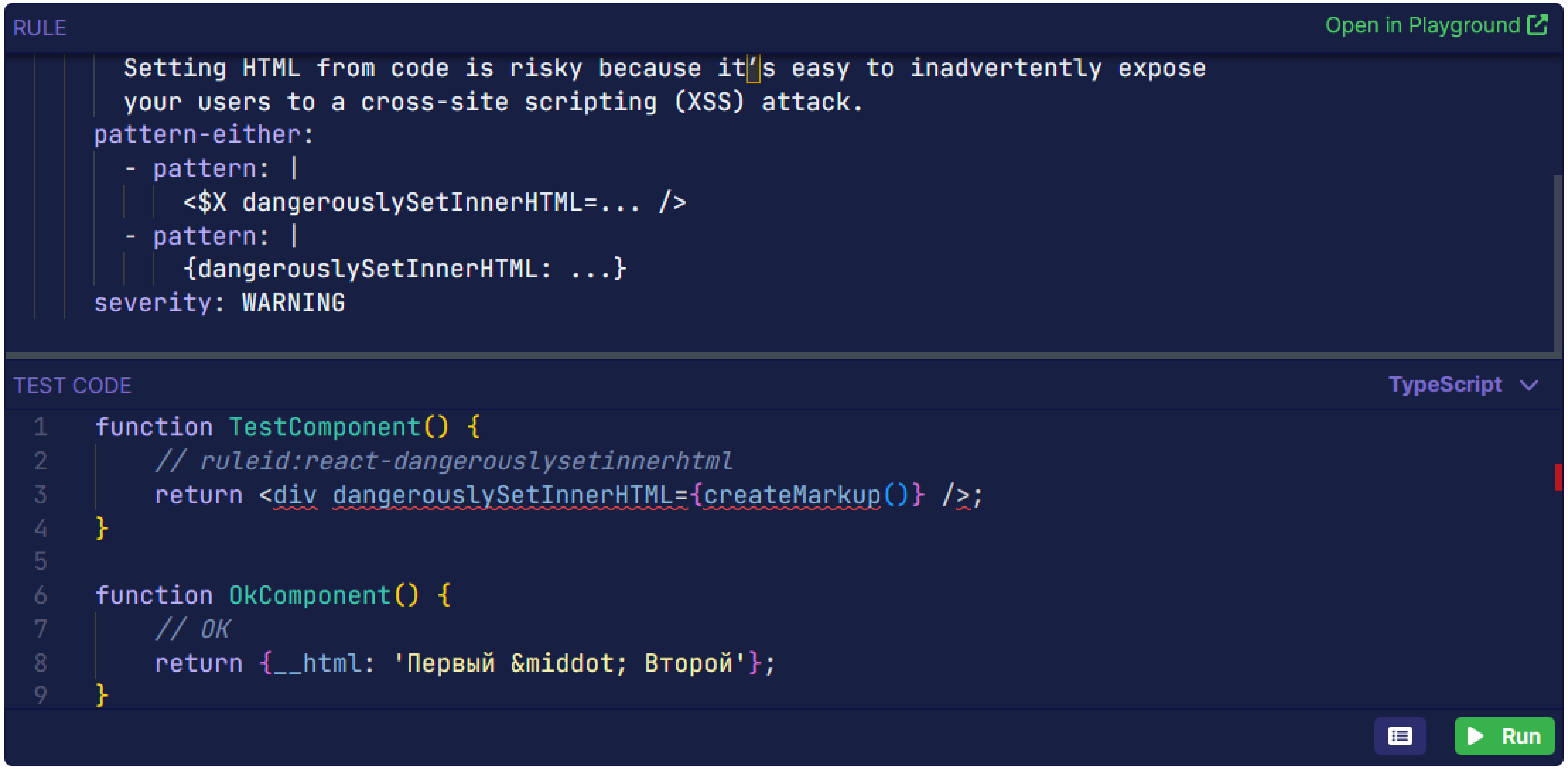
Conditional Operator in Semgrep

simple mode
advanced mode
test code
metadata
docs
Pro Engine beta

```yaml
rules:
  - id: use-of-weak-rsa-key
    message: RSA keys should be at least 2048 bits based on
    NIST recommendation.
    patterns:
      - pattern: |
          KeyPairGenerator $KEY = $G.getInstance("RSA");
          ...
          $KEY.initialize($BITS);
      - metavariable-comparison:
          comparison: $BITS < 2048
          metavariable: $BITS
    languages: [java]
    severity: WARNING
```

```java
import java.security.KeyPairGenerator;

public class WeakRSA {

  static void rsaWeak() {
    // ruleid: use-of-weak-rsa-key
    KeyPairGenerator keyGen = KeyPairGenerator.getInstance
    ("RSA");
    keyGen.initialize(512);
  }

  static void rsaOK() {
    // ok: use-of-weak-rsa-key
    KeyPairGenerator keyGen = KeyPairGenerator.getInstance
    ("RSA");
    keyGen.initialize(2048);
  }
}
```

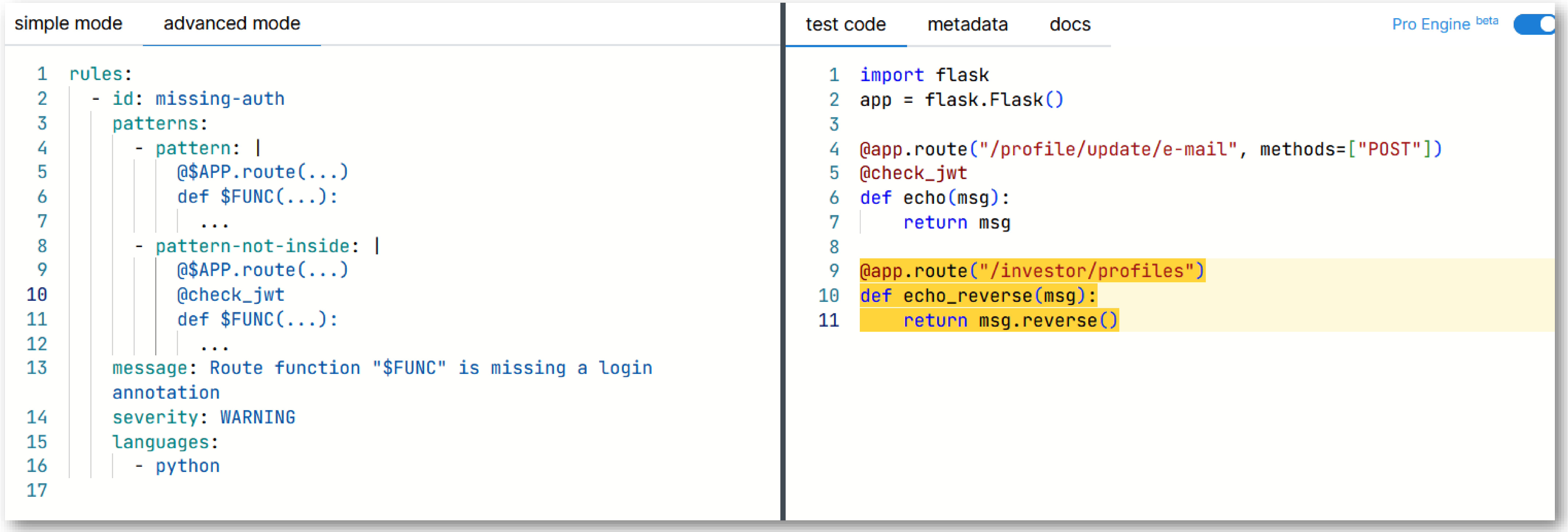
Operators in Semgrep



```
    Setting HTML from code is risky because it’s easy to inadvertently expose
    your users to a cross-site scripting (XSS) attack.
  pattern-either:
    - pattern: |
        <$X dangerouslySetInnerHTML=... />
    - pattern: |
        {dangerouslySetInnerHTML: ...}
  severity: WARNING
```

```
function TestComponent() {
    // ruleid:react-dangerouslysetinnerhtml
    return <div dangerouslySetInnerHTML={createMarkup()} />;
}

function OkComponent() {
    // OK
    return {__html: 'Первый &middot; Второй'};
}
```

Operators in Semgrep



```yaml
rules:
  - id: missing-auth
    patterns:
      - pattern: |
          @$APP.route(...)
          def $FUNC(...):
            ...
      - pattern-not-inside: |
          @$APP.route(...)
          @check_jwt
          def $FUNC(...):
            ...
    message: Route function "$FUNC" is missing a login
    annotation
    severity: WARNING
    languages:
      - python
```

```python
import flask
app = flask.Flask()

@app.route("/profile/update/e-mail", methods=["POST"])
@check_jwt
def echo(msg):
    return msg

@app.route("/investor/profiles")
def echo_reverse(msg):
    return msg.reverse()
```

Credits

https://semgrep.dev/docs/
https://www.flaticon.com/free-icon/spaceship_901787
https://www.flaticon.com/free-icon/spaceship_1114531

THAT'S ALL FOLKS !