



OWASP

Open Web Application
Security Project

DevSecOps

Notre recette pour intégrer la Sécurité du code à un cycle DevOps

sonarsource

Meeting sponsor

Nicolas Bontoux | @nicoallgood | September 2019

OWASP Geneva
Sept. 3rd, 2019



Quick intro

(t'es qui?)



sonarsource

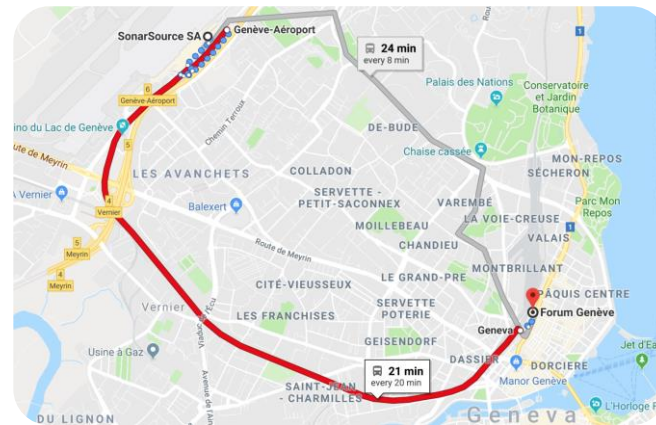
Product Marketing



Support Engineer



Software Developer



De quoi va-t-on parler?



sonarsource 
*Un petit peu
(retour d'expérience)*

Code Quality

Yes!

Code Security

Yes yes!

Code Quality

Selon SonarSource



Maintenabilité

- Faciliter les changements de code
- Augmenter la vélocité
- Avoir la banane
- #dette_technique



Fiabilité

- Tuer les bugs!
- Pas de comportement indéterminé
- #NullPointerException



Code Quality

Selon SonarSource



Maintenabilité

- Faciliter les changements de code
- Augmenter la vélocité
- Avoir la banane
- #dette_technique



Fiabilité

- Tuer les bugs!
- Pas de comportement indéterminé
- #NullPointerException



Sécurité

- Résistant aux attaques
- Pas de fuite de données
- Pas de corruption de données
- #SQL_injection

Security



Application Security

De plus près

SCA

Software Composition
Analysis
(ou Open Source Analysis)

Dépendances vulnérables



...

5,012 commits 53 branches 22 releases 185 contributors MIT

⚠ We found a potential security vulnerability in one of your dependencies. [Dismiss](#)

The **moment** dependency defined in **package-lock.json** has a known **moderate severity** security vulnerability in version range **< 2.19.3** and should be updated.

[Review vulnerable dependency](#)

Only users who have been granted [access to vulnerability alerts](#) for this repository can see this message.
[Learn more about vulnerability alerts](#)

Branch: master New pull request Create new file Upload files Find file Clone or download

Application Security

De plus près

SCA

Software Composition
Analysis
(ou *Open Source Analysis*)

Dépendances vulnérables

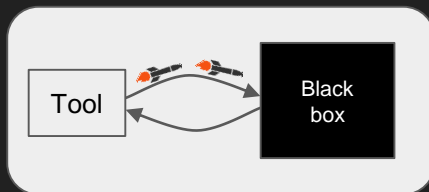


...

DAST

Dynamic Application
Security Testing

Penetration testing



SAST

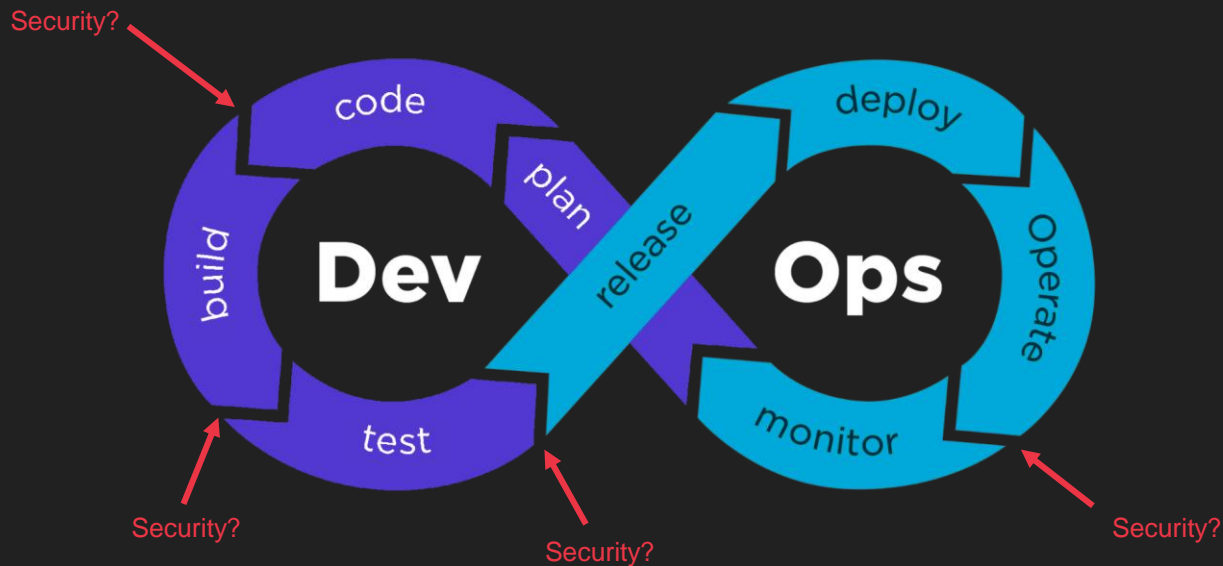
Static Application
Security Testing

Revue de code automatique



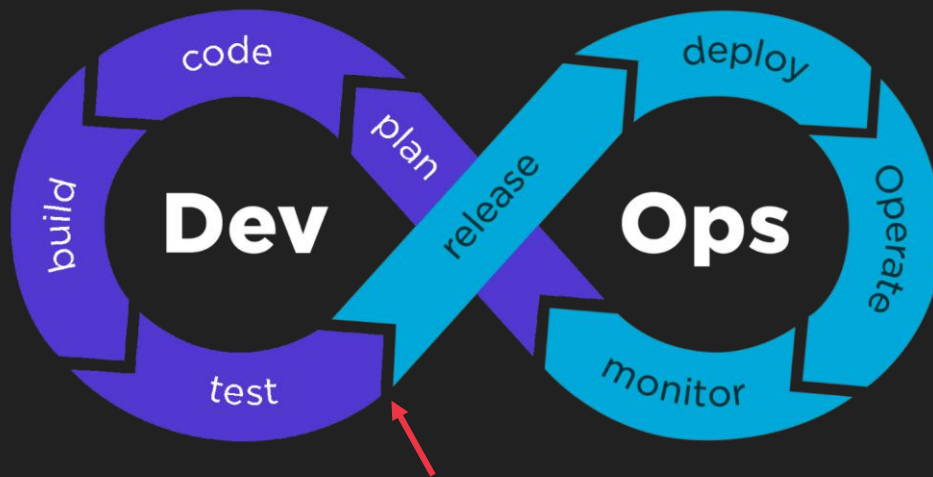
DevSecOps - tentative #1

= DevOps + Security ?



DevSecOps - tentative #1

= DevOps + Security ?

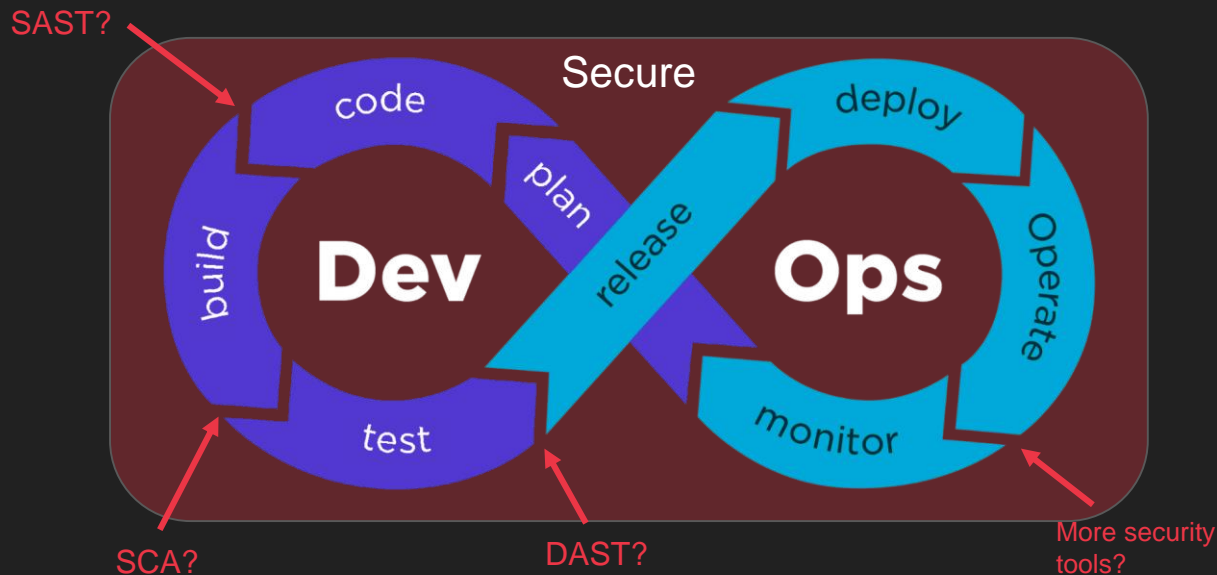


Security!

(car certains outils ont
besoin des binaires)

DevSecOps - tentative #2

= *Secure DevOps*



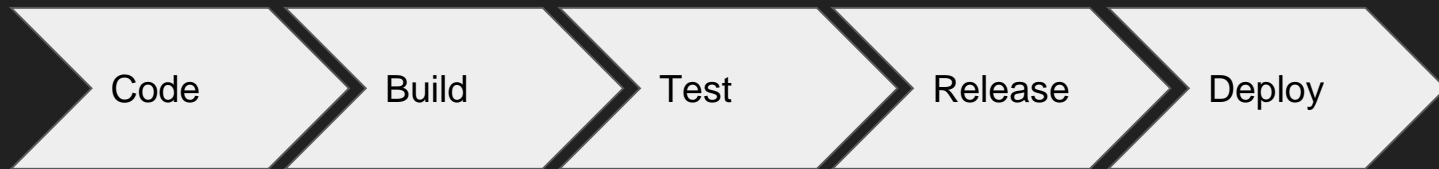
DevSecOps

Le bon état d'esprit

- DevSecOps n'est pas qu'une affaire d'outil(s)
- C'est insuffler tout un état d'esprit 'Sécurité' dans la chaîne CI/CD, c'est redéfinir la notion de 'bon code'
- C'est partager le bon feedback sécurité, au bon moment, à la bonne personne

Shift Left

⚠ buzzword ⚠



Security Testing



A gauche toute!

Code Review

Le juste équilibre

Que du bonheur 😊

Pas de context switching

-

Feedback précis et localisé

-

Opportunité d'apprendre

Les travers 😐

Ca met trop de temps!!

-

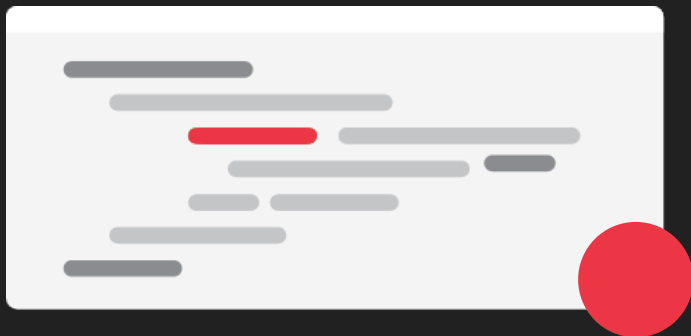
Conseils ≠ Actions

-

Faux positifs

Quiz!

Un peu de Code Review,
tous ensemble.



Action?

ou

Réflexion?

Quiz!



Quiz!



ou



Code Review

En pratique

```
1 ... <?php
2
3 if( isset( $_GET[ 'Submit' ] ) ) {
4     // Get input
5     1 $id = $_GET[ 'id' ];
6
7     // Check database
8     3 $getid = 2 "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
9     $result = 4 mysql_query($GLOBALS["__mysqli_ston"], $getid );
```

Refactor this code to not construct SQL queries directly from tainted user-controlled data. ...

5 months ago ▾ L9 🔗

🔒 Vulnerability ▾ ! Blocker ▾ 🟢 Confirmed ▾ Not assigned ▾ 30min effort Comment

🔍 cert, cwe, owasp-a1, sans-top25-inse... ▾



Action!

Code Review

En pratique

```
435 public static KeyPair generateRsaOrDsa(boolean rsa) throws Exception {  
436     if (rsa) {  
437         KeyPairGenerator keyPairGen =  
438             KeyPairGenerator.getInstance("RSA");  
439         keyPairGen.initialize(1024);
```

9 years ago ▾ L439 🔗

🔗 Open ▾ Not assigned ▾ 2min effort Comment

🔖 cwe, owasp-a3 ▾





Action!

Code Review

En pratique

```
102  
103 public Cookie build() {  
104     Cookie cookie = new Cookie(requireNonNull(name), value);
```

 Open

9 months ago ▾ L104 

 cert, cwe, owasp-a3

```
105     cookie.setPath(getContextPath(request));  
106     cookie.setSecure(isHttps(request));
```



Réflexion

Code Review

En pratique

```
39     EntityManager em;  
40  
41     protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {  
42         String name = req.getParameter("name");  
43         String[] array = new String[] {name, "abc"};  
44  
45         em.createQuery(array[0]);  
46         em.createQuery(array[1]);  
47     }
```



Action!

Retour d'expérience

Pragmatic DevSecOps

Tout n'est pas automatisable

-

Vulnérabilité \neq Code à risque

-

Commencez par la sécurité du code

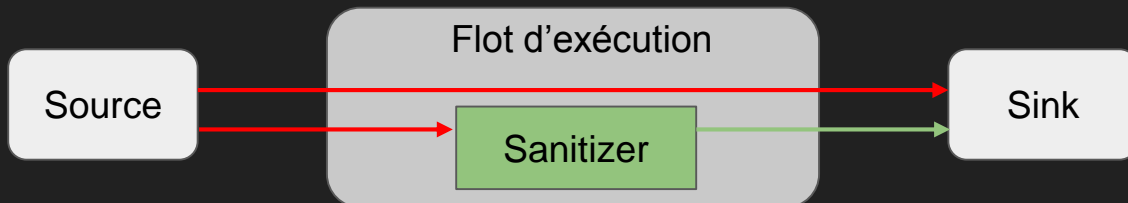
-

Il faut pouvoir comprendre, et apprendre

Bonus

SAST

Taint analysis



Source

```
1  ... <?php
2
3  $conn = new PDO('mysql:host=localhost;dbname=prod', $user, $pass);
4
5  if (isset($_POST['email'])) {
6
7      $email = $_POST['email'];
8
9      $sql = "SELECT * FROM USERS WHERE email = " . $email . " ";
10     $user = $conn->query($sql);
11
12     if ($user->rowCount) {
13         // do something
14     }
```

Refactor this code to not construct SQL queries directly from tainted user-controlled data. ...

3 days ago ▾ L10 🔗

🔒 Vulnerability 🚫 Blocker 🔵 Open Not assigned 30min effort

🔍 cert, cwe, owasp-a1, sans-top25-inse...

Sink
#badaboum

Bonus

Security Hotspots

2 types of security issues

**Security
Hotspot**

Code Review

**Security
Vulnerability**

Code Fix

Bonus

User Experience

PR + Taint Analysis (Java, PHP, JS) src/main/java/devoxx/security/injection/Servlet.java

```
26 ... }
27
28 2 String name = 1 taintedRequest.getParameter(FIELD_N
29 3 taintedString = name;
30
31 taintedString = 4 java.net.URLDecoder.decode(tainte
32
33 try {
34 5 new SQLInjectionVulnerability(taintedString);
35 new CommandInjectionVulnerability(taintedString);
36 new RegexInjectionVulnerability(taintedString);
```

Refactor this code to not construct SQL queries directly from tainted user-controlled data.

🔒 Vulnerability +6

Servlet.java

- 1 source: taint value is propagated
- 2 taint value is propagated
- 3 taint value is propagated
- 4 taint value is propagated
- 5 taint value is propagated

SQLInjectionVulnerability.java

- 6 sink: taint value is propagated

alt + ↑ ↓ to navigate issue locations

Refactor this code to not construct SQL queries directly from tainted user-controlled data. See Rule 3 months ago L18

🔒 Vulnerability 🚫 Blocker 🔓 Open Not assigned 30min effort cert, cwe, owasp-a1, sans-top25-inse...

```
16 ... }
17
18 stmt.execute( 6 SELECT_SQL + uname);
19
20
21 stmt.execute("select FNAME, LNAME, SSN from USERS where UNAME = default");
22 }
23 }
```

Bonus

Rules

The screenshot displays the SonarCloud web interface. At the top, the browser address bar shows the URL: `https://sonarcloud.io/project/issues?id=sonarsource-devovx-demo_devovx-demo&open=AWnZJaVa2ZTYZ-vhhMty&resolved=fals...`. The SonarCloud header includes navigation links for 'My Projects' and 'My Issues', a search bar, and a user profile icon. The main content area is for the project 'sonarsource-devovx-demo' and the issue 'PR + Taint Analysis (Java, PHP, JS)'. A sidebar on the left lists various issues, including 'Vulnerability' and 'Use a logger to log this exception.'. The central pane shows a code snippet from `src/main/php/security/sql-single-file.php` with a highlighted vulnerability: 'Refactor this code to not construct SQL queries directly from tainted user-controlled data.' The code snippet is as follows:

```
3 -  
4 if (isset($_POST['email'])) {  
5     $email = $_POST['email'];  
6  
7     //sql = "SELECT * FROM USERS WHERE email = '" . $email . "'";  
8     $sql = "SELECT * FROM USERS WHERE email = '$email'";  
9  
10    $user = $conn->query($sql);
```

Below the code, a detailed description of the rule is provided:

Database queries should not be vulnerable to injection attacks

Vulnerability **Blocker** **Main sources** **cwe, owasp-a1, sans-top25-insecure, ...** Available Since Mar 26, 2019 Security SonarAnalyzer (PHP) Constant/Issue: 30min

User provided data, such as URL parameters, should always be considered untrusted and tainted. Constructing SQL queries directly from tainted data enables attackers to inject specially crafted values that change the initial meaning of the query itself. Successful SQL injection attacks can read, modify, or delete sensitive information from the database and sometimes even shut it down or execute arbitrary operating system commands.

Typically, the solution is to rely on the prepared statements rather than string concatenation, which ensures that user provided data will be properly escaped.

This rule supports: Native Database Extensions, PDO, Symfony/Doctrine, Laravel/Eloquent.

Noncompliant Code Example

```
function authenticate() {  
    if( isset( $_POST[ 'Connect' ] ) ) {  
        $login = $_POST[ 'login' ];  
        $pass = $_POST[ 'pass' ];
```

