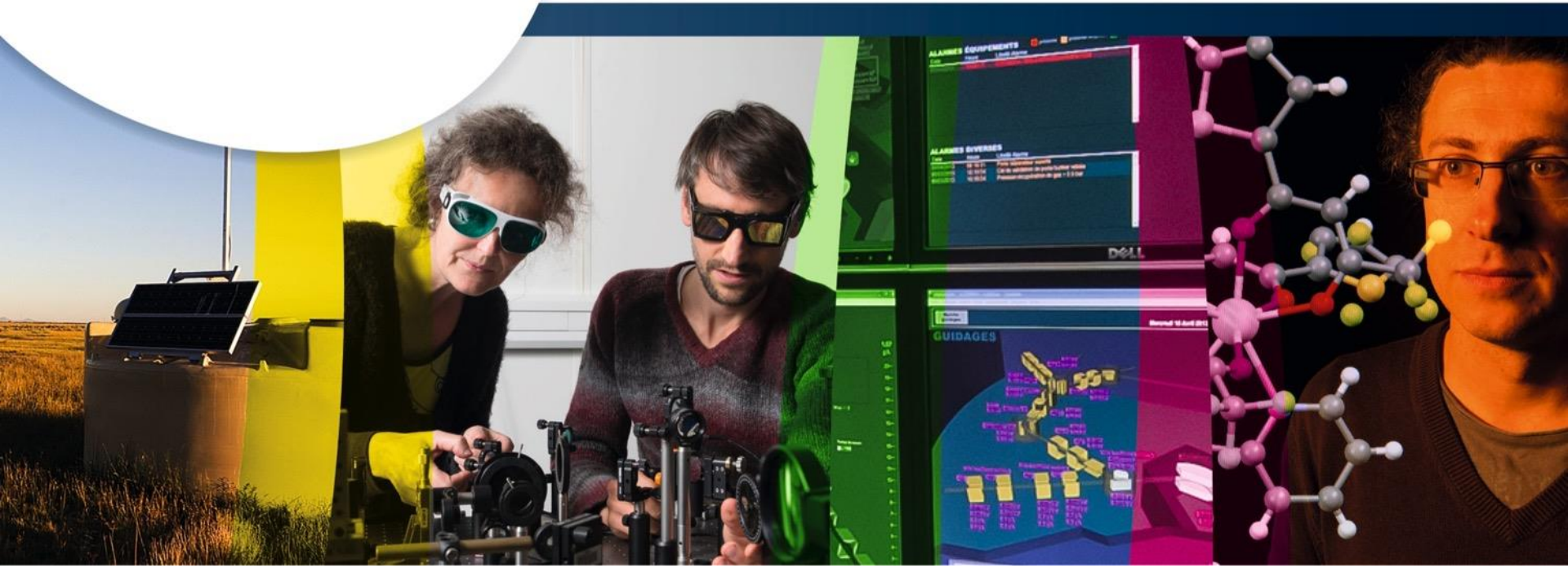
www.cnrs.fr

Organiser et mettre en œuvre la sécurité dans les applications Web

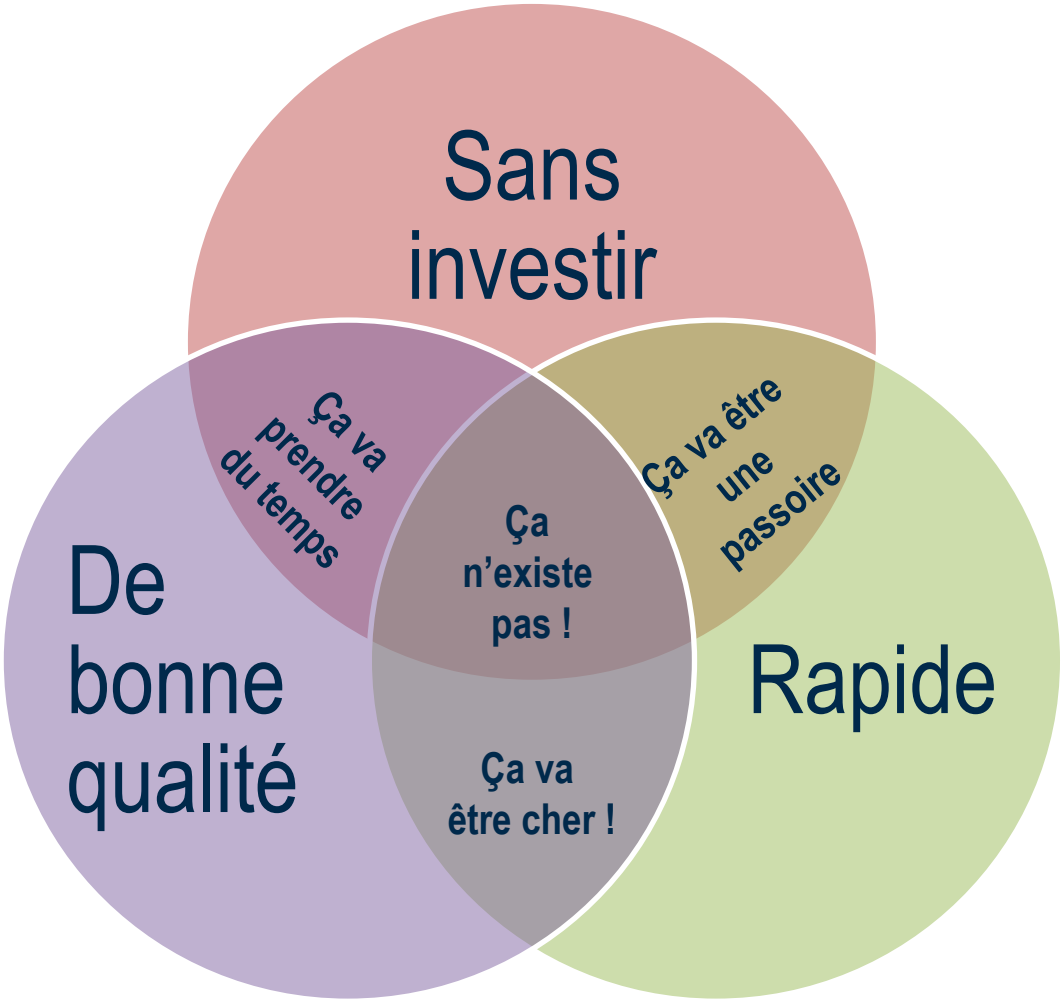
Bonnes pratiques



Pratiques de développement

www.cnrs.fr

La fin des illusions



Sans qualité, pas de sécurité

www.cnrs.fr

- Qualité logicielle
  - Capacité Fonctionnelle de ISO/CEI 9126
- Comment favoriser la qualité pour favoriser la sécurité ?
  - Maîtriser l'écosystème
    - Langages de programmation
    - Quelques fondamentaux
  - Evaluer ses compétences
  - Comprendre ce que nous faisons
  - Comprendre ce que le code fait réellement
  - Se donner les moyens

www.cnrs.fr

Maîtriser l'écosystème

Ecosystème à la Prévert
Oh, une fiche de poste …

- **Langages**
  - Backend : PHP, Python, Java, Node.js, …
  - frontend : PHP, HTML, CSS, Less, Sass, javascript, …
- **Frameworks**
  - backend, frontend
- **Moteurs de production**
  - Maven, grunt, gulp, gradle, ant, npm, …
- **Environnement de développement intégré**
  - Eclipse, IntelliJ, phpStorm, SublimeText, vi, Emacs, …
- **Gestionnaires de version**
  - Git, Svn, Mercurial
- **Outils de forge qualité**
  - Jenkins, Sonar, nexus, …
- **Bases de données**
  - SQL, NoSQL
- **Serveurs**
  - Apache, tomcat, Nginx
- **Méthodologies**

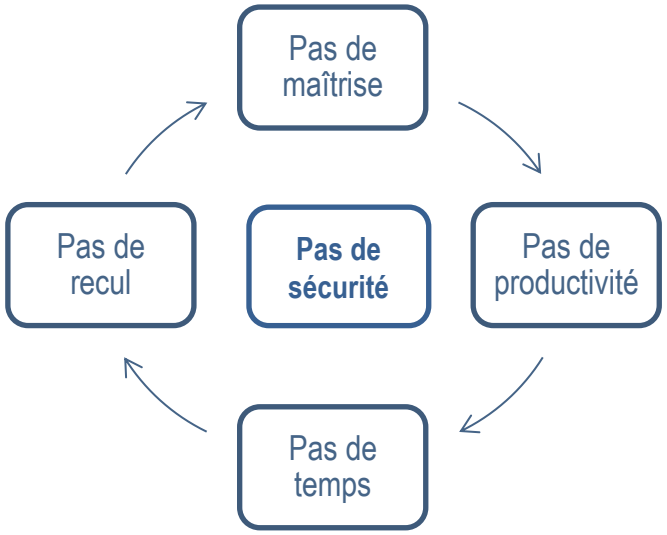
- Un peu de réseau,
- Un peu d'administration système
  - Windows, Linux,
  - Docker, Vagrant
- …

- Sans oublier…
  - Outils de bureautique
  - Communication

Maîtriser l'écosystème

www.cnrs.fr

- Sans un minimum de maîtrise, pas de sécurisation probante



www.cnrs.fr

Maîtriser son langage de programmation

Maîtriser son langage de programmation

www.cnrs.fr

- Le **bon langage**, c'est celui que l'on **maîtrise**
  - ***Facilité d'usage***
    - produire du code aisément
    - sans avoir à descendre trop bas (allocation, pointeurs)
  - ***Clarté***
    - le langage est compréhensible et n'est pas trompeur (*error-prone*)
  - ***Performance***
  - ***Qualité de son écosystème***
    - **Librairies & frameworks** : ne pas avoir à tout réinventer
    - **Outillage** : serveurs, éditeurs : produire sur de bonnes bases
  - ***Support***: communauté, forum
- **Sécurité**: suivi, réactivité, vulnérabilités patchées, …

Maîtriser son langage de programmation: points d'attention

- Faire l'effort de *vraiment* l'apprendre
  - "*Javascript, tout le monde l'utilise sans jamais l'avoir appris*"
  - Autrement que par des tutoriaux douteux…
- Être conscient de ses **particularismes**, **forces** et **faiblesses**
  - Sureté de typage
  - Encapsulation
  - Manipulation de la vérité et des nombres
  - Facilités de refactoring

Maîtriser son langage de programmation:

Sureté de typage

- Typage
  - Classifier les objets et définir les opérations possibles selon la classe de l'objet
    - *"Ne pas additionner des choux et des carottes"*
- Différents typages
  - **Statique** : le type est défini avant utilisation et vérifié à la compilation
  - **Dynamique** : le type est associé à l'exécution
  - **Explicite**: pas de conversion implicite entre types sans relation
  - **Faible** : conversion implicite possible entre types de nature différente

- **Sureté de typage**
  - *"well typed programs do not go wrong"*
  - S'assurer qu'un type est ***conforme*** aux attendus
  - Rejeter les opérations ***absurdes*** (ou malicieuses)

Sureté de fonctionnement

- Faire passer un chou pour une carotte est une vulnérabilité
- Si votre langage ne *renforce* pas la vérification de type, **c'est à vous de le faire**
  - Pour les paramètres en entrées
  - Pour les valeurs de sortie
- Histoire des 3 petits cochons…

Vérification de type en entrée

https://repl.it/FKVu/3

PHP

```php
class Pig {
  function isOk() {
    return true;
  }
}

class Wolf {
  function isOk() {
    return true;
  }
}

// La maison des 3 petits cochons
class PigHouse {
  function openTheDoor($pig) {
    if ( $pig->isOk()) {
      echo "Enter ".get_class($pig)."\n";
    }
  }
}

$house = new PigHouse();
$pig = new Pig();
$wolf = new Wolf();

$house->openTheDoor($pig);  // Enter Pig
$house->openTheDoor($wolf); // Enter Wolf
```

Pas de typage

Typage

```php
class PigHouse {
  function openTheDoor(Pig $pig) {
    if ( $pig->isOk()) {
      echo "Enter ".get_class($pig)."\n";
    }
  }
}

//...
```

```
Uncaught TypeError: Argument 1 passed
to PigHouse::openTheDoor() must be an
instance of Pig, instance of Wolf given
```

Vérification de type en sortie

https://repl.it/FKXm/4


```php
class Pig{}

class Wolf{}

$list = array( new Pig(), new Wolf(), new Pig());


class PigList {
  function __construct ($list) {
    $this->list = $list;
  }
  function nextPig() {
    return array_shift($this->list);
  }
}

$pigList = new PigList($list);


while( $pig = $pigList->nextPig()) {
  echo "Enter Mr. ".get_class($pig)." \n";
}
// Enter Mr. Pig
// Enter Mr. Wolf
// Enter Mr. Pig
```

Pas de typage

Vérification de typage… à la main

```php
while( $pig = $pigList->nextPig()) {
  if ( $pig instanceof Pig ) {
    echo "Enter Mr. ".get_class($pig)." \n";
  }
}
```

Vérification de type à la compilation

https://repl.it/FK0A/5

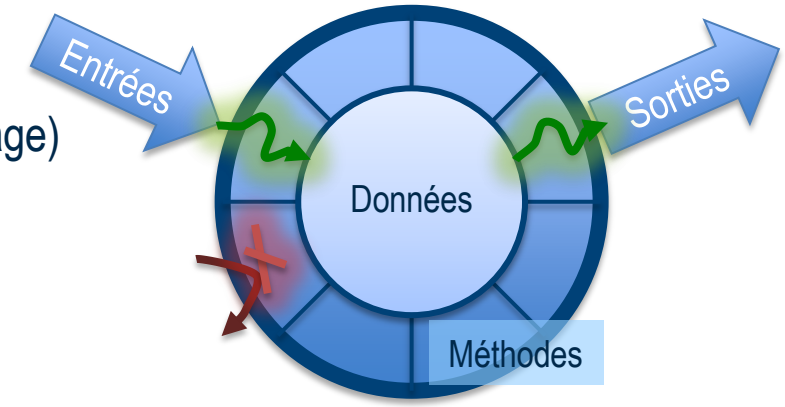
```
class Main {
  public static void main(String[] args) {

    Pig pig = new Pig();
    Wolf wolf = new Wolf();

    openTheDoor(pig);
    openTheDoor(wolf);
  }

  public static void openTheDoor(Pig pig) {
    System.out.println("Enter Mr. "+pig.getClass());
  }
}
```

Main.java:7: error: incompatible types:
Wolf cannot be converted to Pig openTheDoor(wolf);

Encapsulation

www.cnrs.fr

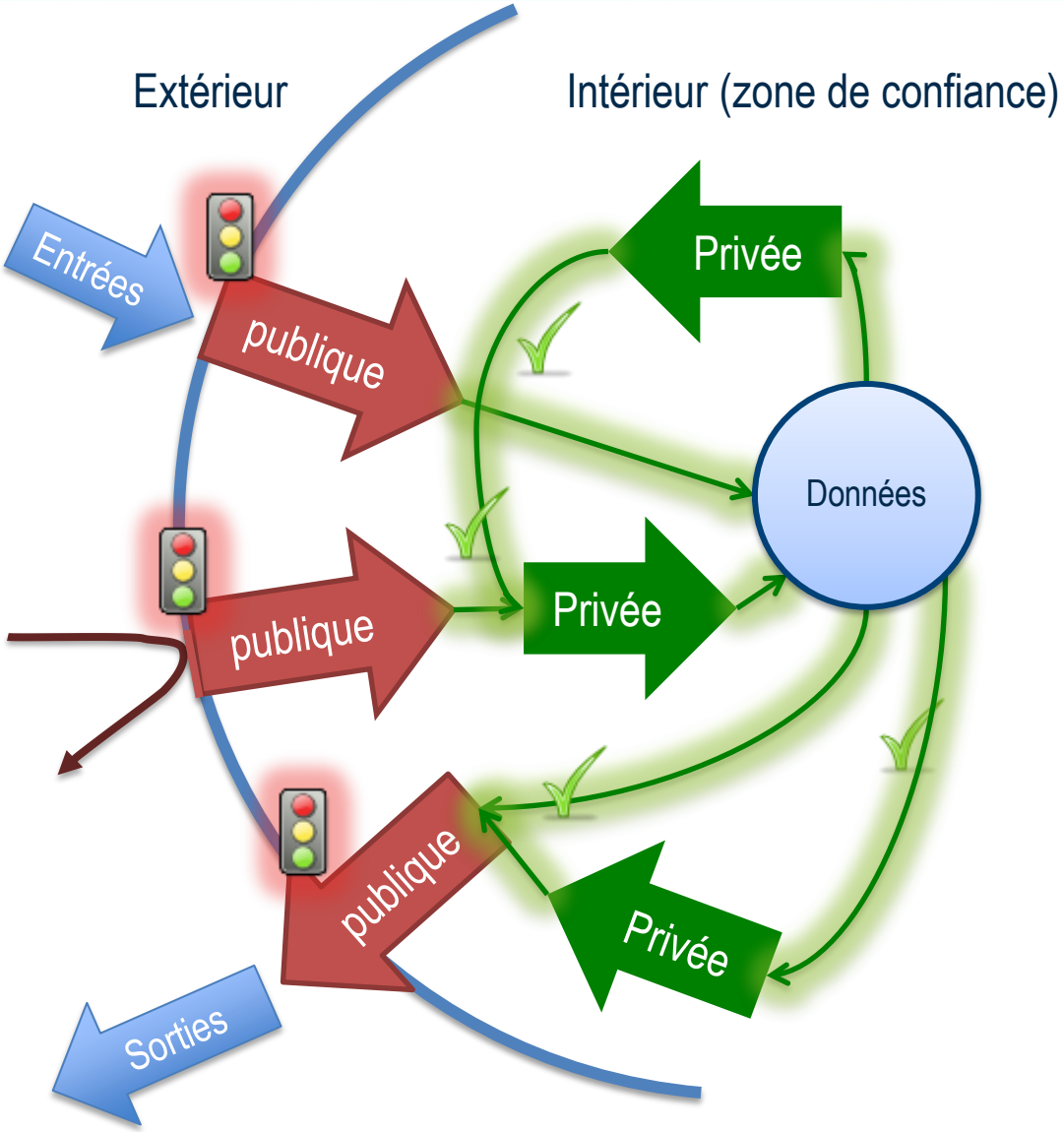
- **Protéger** les données et n'y accéder qu'au travers de **méthodes**
- Pilier de la programmation objet



- Niveaux de protection (suivant langage)
  - Privé (interne)
  - Protégé (communauté)
  - Public

- **Niveau de protection ⇔ niveau de vigilance ⇔surface d'attaque**

Niveau de protection



Maîtriser son langage de programmation : particularismes : la vérité est ailleurs

- Les évaluations booléennes ne sont pas uniformes
  - A la grande joie des attaquants…
- Attention aux évaluations *conciliantes*
  - *Conversions de type implicites*
- Tous les signes *égal* ne sont pas *égaux* :
  - `'=='` vs `'==='`
- L'absence *n'est pas* une évidence
  - `null, none, undefined, 1, 0, …`
- En cas de doute, *faire un test tout de suite*
  - En mode interpréteur ou sur https://repl.it/

```
0:        false
42:       true
0.0:      false
4.2:      true
"":       false
"string": true
"0":      false
"1":      true
[1, 2]:   true
[]:       false
stdClass: true
```

PHP

```
''         ==   '0'          //false
0          ==   ''           //true
0          ==   '0'          //true
false      ==   'false'      //false
false      ==   '0'          //true
false      ==   undefined    //false
false      ==   null         //false
null       ==   undefined    //true
" \t\r\n"  ==   0            //true
```

javascript

Quand == <> ===

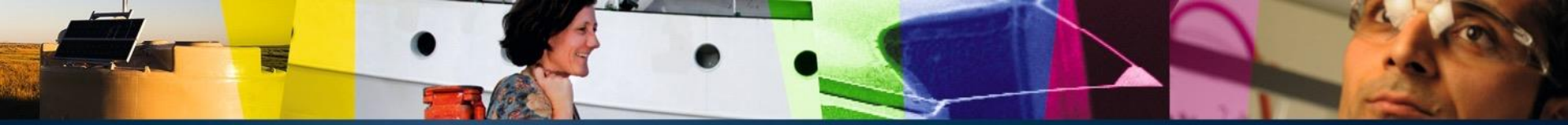
PHP

Comparaison large avec ==

	TRUE	FALSE	1	0	-1	"1"	"0"	"-1"	NULL	array()	"php"	""
TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE
1	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
0	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE
-1	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
"1"	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"0"	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
"-1"	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
NULL	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE
array()	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
"php"	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
""	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE

Comparaison stricte avec ===

	TRUE	FALSE	1	0	-1	"1"	"0"	"-1"	NULL	array()	"php"	""
TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
1	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
0	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
-1	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"1"	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"0"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
"-1"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
NULL	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
array()	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
"php"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
""	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE



Connaître le côté obscur

www.cnrs.fr

- Prendre le temps d'identifier les faiblesses et incohérences du langage
  - **"Lisez le sacré manuel" !**
  - https://stackoverflow.com/search?q=pitfalls
- Construisez une base culturelle commune
  - Faites une Cheat Sheet (ou feuille de triche ☺) *Zak' si tu nous entends*
  - Se tester mutuellement

- Quelques références

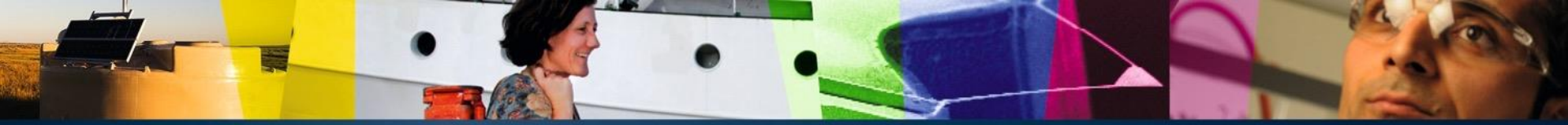
  Java: https://www.securecoding.cert.org/confluence/display/java/Java+Coding+Guidelines

  Python (Owasp PySec): https://github.com/ebranca/owasp-pysec/wiki

  PHP : https://www.owasp.org/index.php/PHP_Security_Cheat_Sheet
- Voir https://www.owasp.org/index.php/Cheat_Sheets

Maîtriser quelques fondamentaux

www.cnrs.fr



Maîtrisez quelques fondamentaux : Algèbre de Boole

- L'erreur de logique binaire est une clef offerte à l'attaquant
- Tout le monde *croit* maîtriser la logique binaire
  - Mais maîtrisez vous *vraiment* l'algèbre de Boole ?

$$a \&\& (a \,||\, b) = a = a.(a + b) = a.a + a.b = a .(1+ b) = a .1 = a$$

$$(a \,||\, b) \&\& (a \,||\, !b) = a = a.a + a.\underline{b} + b.a + b.\underline{b}$$
$$= a + a.\underline{b} + a.b$$
$$= a.(1 + b + \underline{b}) = a$$

Faite donc un peu les exercices
https://fr.wikiversity.org/wiki/Logique_de_base/Exercices/Alg%C3%A8bre_de_Boole
Et situez vous sur l'échelle de Dreyfuss…

Maîtrisez quelques fondamentaux : Algèbre de Boole

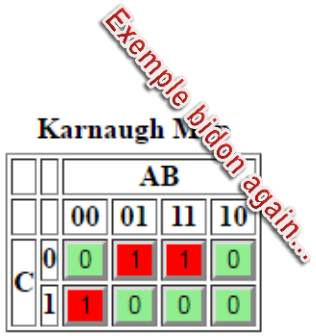
- Utilisez toujours des expressions ou des structurations signifiantes
  - Pour les autres développeurs
  - Pour la MOA

- En cas de doute, faites des tables de vérités

Chercheur	Invité	Manipulateur	Accès Incubateur
F	F	F	F
F	F	V	F
F	V	F	F
F	V	V	F
V	V	V	V

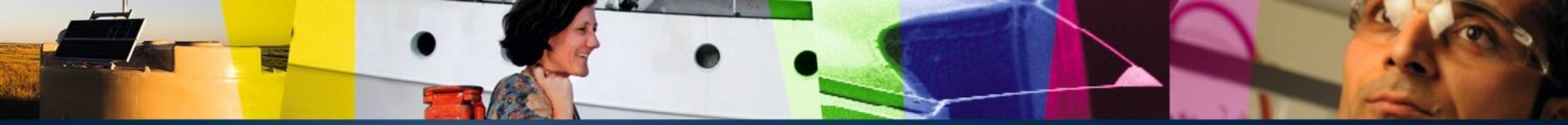
Exemple bidon...

- Réduisez les expressions avec des tables de Karnaugh
  - https://fr.wikipedia.org/wiki/Table_de_Karnaugh
- Utilisez des solveurs pour simplifier ou vérifier
  - http://www.32x8.com/var3.html



Exemple bidon again...

$$F(ABC) = B\,\overline{C} + \overline{A}\,\overline{B}\,C$$



Maîtrisez quelques fondamentaux :
Les Design Patterns

www.cnrs.fr

- S'appuyer sur les expériences des prédécesseurs
  - Collection de solutions éprouvées
  - Un contexte ➔ Un problème ➔Une solution
- Apprendre en observant
  - Ne pas perdre de temps à retrouver une solution connue

- Patrons de conception classiques (Gang Of Four)
  - *Création, Structure, Comportement*
- Patrons de responsabilité GRASP
  - *General Responsibility Assignment Software Patterns/Principles*

Merci patron…

- Patrons de création
  - Singleton
  - Prototype
  - Fabrique
  - Fabrique abstraite
  - Monteur
- Patrons de structure
  - Pont
  - Façade
  - Adaptateur
  - Objet composite
  - Proxy
  - Poids-mouche
  - Décorateur
- Patrons de comportement
  - Chaîne de responsabilité
  - Commande
  - Interpréteur
  - Itérateur
  - Médiateur
  - Mémento
  - Observateur
  - État
  - Stratégie
  - Patron de méthode
  - Visiteur
- Patrons GRASP
  - Expert en information
  - Créateur
  - Faible couplage
  - Forte cohésion
  - Contrôleur
  - Polymorphisme
  - Fabrication pure
  - Indirection
  - Protection

Design patterns:
Exemple

- **Contexte**
  - Instancier une classe une seule fois
  - Accès à l'instance à divers endroits
- **Exemple**
  - Accès à un dictionnaire de configuration
- **Solution naïve**
  - Passage en paramètre à chaque fois qu'on doit l'utiliser
- **Problème**
  - Difficile à maintenir
- **Solution**
  - Pattern du **singleton**
    - Constructeur privé
    - Instance en tant qu'attribut statique de la classe
    - Méthode statique d'accès à l'instance

Identifier les anti-patterns

www.cnrs.fr

- **Un anti-pattern, c'est comme un pattern, il donne une solution qui ressemble à une solution, mais qui n'en est pas une.**
- S'appuyer sur les expériences *malheureuses* des prédécesseurs
- Les attaquants adorent les anti-patterns
  - Vous pensez avoir une solution
  - L'attaquant voit le défaut
- Exemples
  - *''God Object"* : concentrer toutes les responsabilités dans une classe
  - *''Negative Cache"* : garder en cache les erreurs de traitement

www.cnrs.fr

Comprendre ce que le code fait

Le complexe de Frankenstein

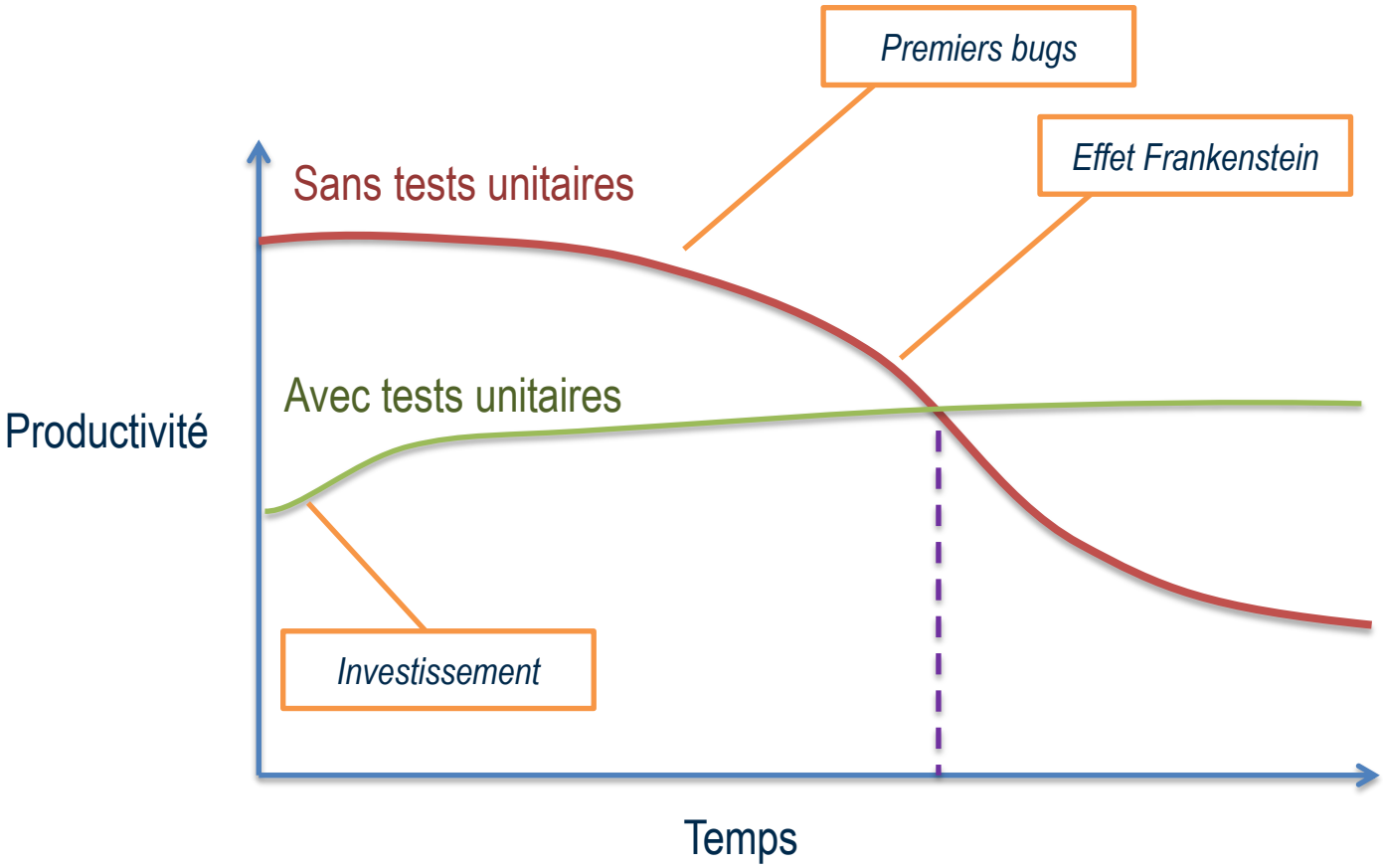
- Quand est-ce que votre création vous dépasse ?
  - Quand elle est tombée en marche
  - Quand vous ne comprenez plus vraiment ce qui se passe
  - Quand vous avez peur des effets de bords
- **Aucune** vision crédible du respect des exigences de sécurité
- Pour garder le contrôle
  - Une conception et des exigences claires
  - Usage de Designs Pattern identifiables
  - Utilisez les tests unitaires

Les Tests Unitaires Automatisés (xUnit)

www.cnrs.fr

- *Il n'y a plus d'excuses* pour ne pas faire de tests unitaires en 2017
  - Des librairies dans tous les langages !
  - Un investissement vite rentabilisé et une productivité constante
- **Apports des tests unitaires**
  - Une confiance objective : la portion de code a été exécutée
  - Un filet de sécurité permanent
  - Un capitale croissant : les tests unitaires s'accumulent
  - Une incitation à mieux concevoir : imaginer le test aide à créer la classe
- **L'automatisation** permet d'envisager un refactoring avec sérénité
  - ➔ Frankenstein à la niche !
  - ➔David Dageot https://www.youtube.com/watch?v=q11gydDAMSo

Courbe (idéalisée) de productivité avec tests unitaires



Tester, tester, tester…

www.cnrs.fr

- Tester les portions **importantes**, **critiques**
- Tester les conditions *nominales*, *extrêmes* et *inattendues* (sortir du cadre)

- Exemple: création d'une adresse mail unique

```
def mail(nom, prenom, unite, domain)
```

  - L'adresse est créée pour la première fois (cas nominal)
  - L'adresse **n'est pas** créée quand
    - le nom ou le prénom ou l'unité ou le domaine sont vides, nuls, des entiers
    - l'adresse existe déjà
    - le nom est celui d'une adresse existante *plus un blanc*
    - la base de compte ne répond pas
    - …
- Tester implique déjà de penser à la conception avant même de coder ☺

www.cnrs.fr

(re)Connaître ses (in)compétences

(re)Connaître ses (in)compétences

www.cnrs.fr

- Les attaques profitent de nos erreurs
- Dilution de l'expertise et accroissement du périmètre
  - Notre champ d'action est de plus en plus vaste
  - "Un grand pouvoir implique de grandes responsabilités" – Spiderman
- L'ennemi, c'est le *"tombe-en-marche"*
- Pour être capable de limiter les défaillances,
  - Identifier nos compétences par domaine,
    - Monter en compétence sur les domaines critiques
    - Ou trouver de l'aide auprès de personnes compétentes

Comment ne pas monter en compétence
Le développeur full-stack… overflow

www.cnrs.fr

- Développeur *full-stackoverflow*
  - Stackoverflow: communauté méritocratique de Question Réponse
  - Construit par agrégation de recettes ⇔magie
  - *"I'm a good programmer or a good googler ?"*

- Ne **jamais** recopier **bêtement** une solution qui *"marche"*
  - Aucune compréhension ➔ *'Novice'*
  - Une solution qui marche n'est pas forcément la bonne
  - L'urgence est mauvaise conseillère

- **Comprendre** ce qui va aller en production !
  - **Demain il faudra vous justifier**

Evaluation du périmètre

Domaine	Novice	Débutant confirmé	Compétent	Efficace	Expert
HTTP		À renforcer			
PHP					
Symfony					
Utilisation SQL					
Admin. MySQL					
CSS					
Javascript					
JQuery					
Admin Linux					
Apache					
Top 10 OWASP		À renforcer			

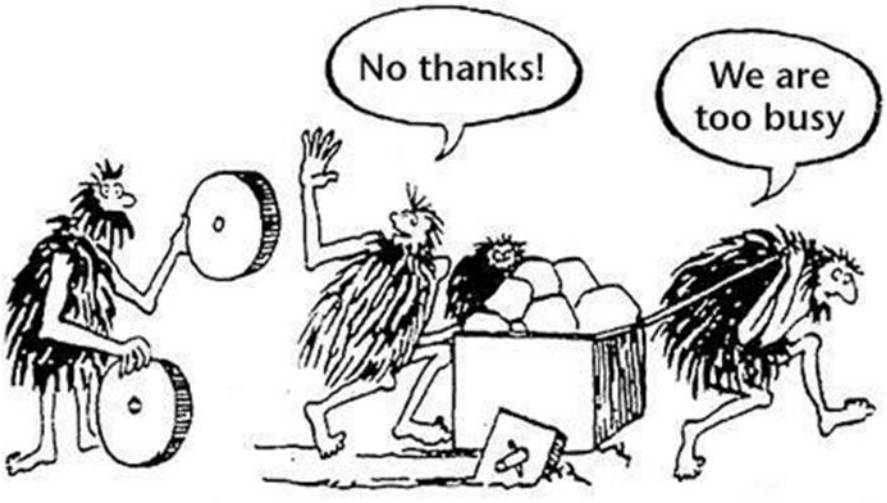
Sait ce qu'il fait

Attention, erreurs probables

www.cnrs.fr

Se donner les moyens

Le vrai problème…



www.cnrs.fr

Maîtriser son environnement de développement

Environnement de Développement Intégré (IDE)

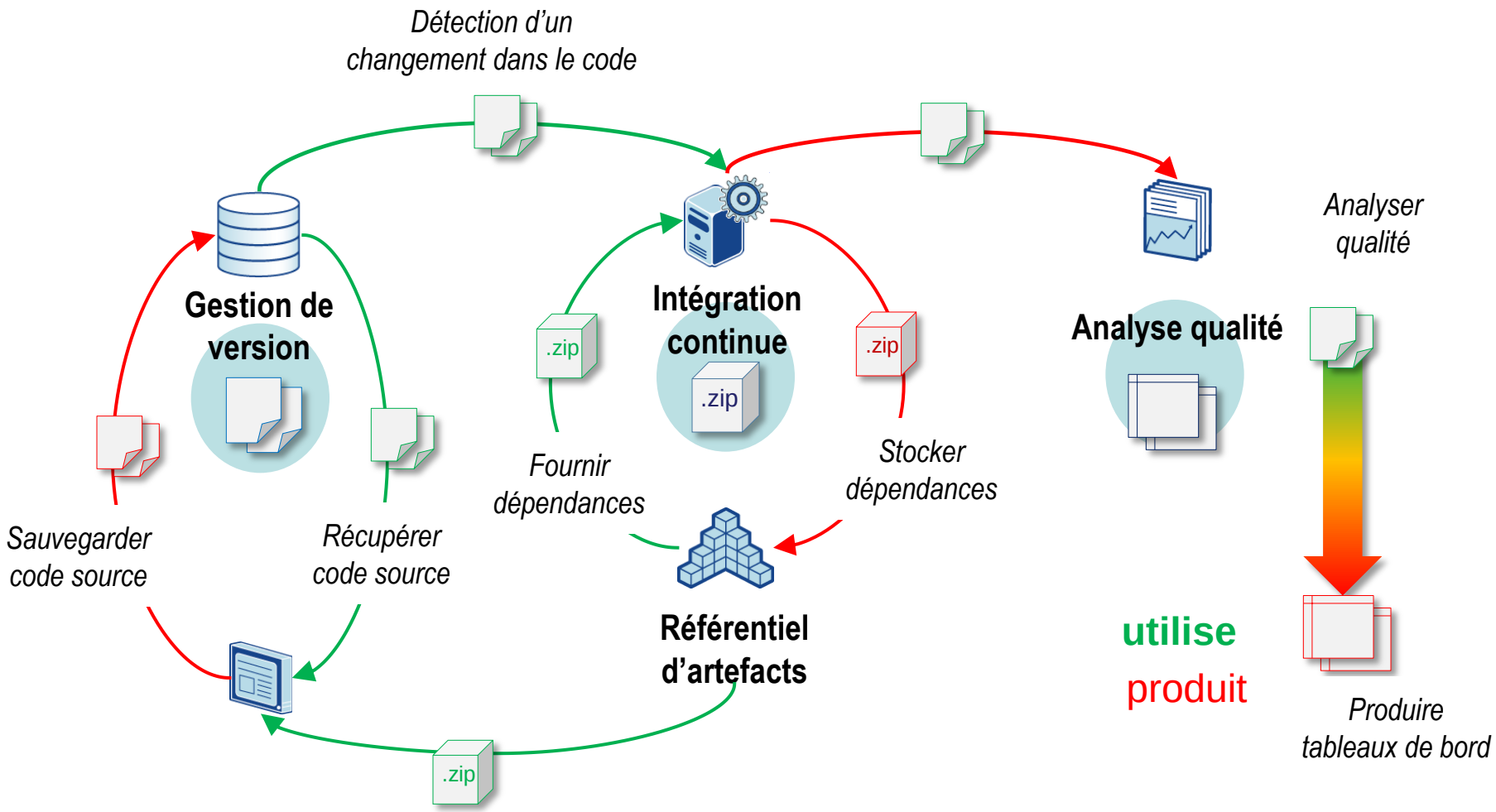
www.cnrs.fr

- C'est **beaucoup plus** qu'un éditeur de texte amélioré
  - Représentation graphique ➔ meilleure compréhension cognitive
  - Automatisation et feedback
    - *taches → action → processus → visualisation → correction*
  - Débogage, profiling, gestion des dépendances, …

- **Apprenez à vous en servir**
  - Tutoriaux, communauté, FAQ
  - Apprenez les raccourcis clavier **(productivité x 10 !)**

- Ne sombrez pas dans l'**auto-magique** ← *'novice'*
  - Comprendre les mécanismes en jeu ← *'compétent'*
    - compilation, packaging, …

Forge


Détection d'un changement dans le code
Gestion de version
Intégration continue
Analyse qualité
Analyser qualité
.zip
.zip
.zip
.zip
Fournir dépendances
Stocker dépendances
Sauvegarder code source
Récupérer code source
Référentiel d'artefacts
utilise
produit
Produire tableaux de bord

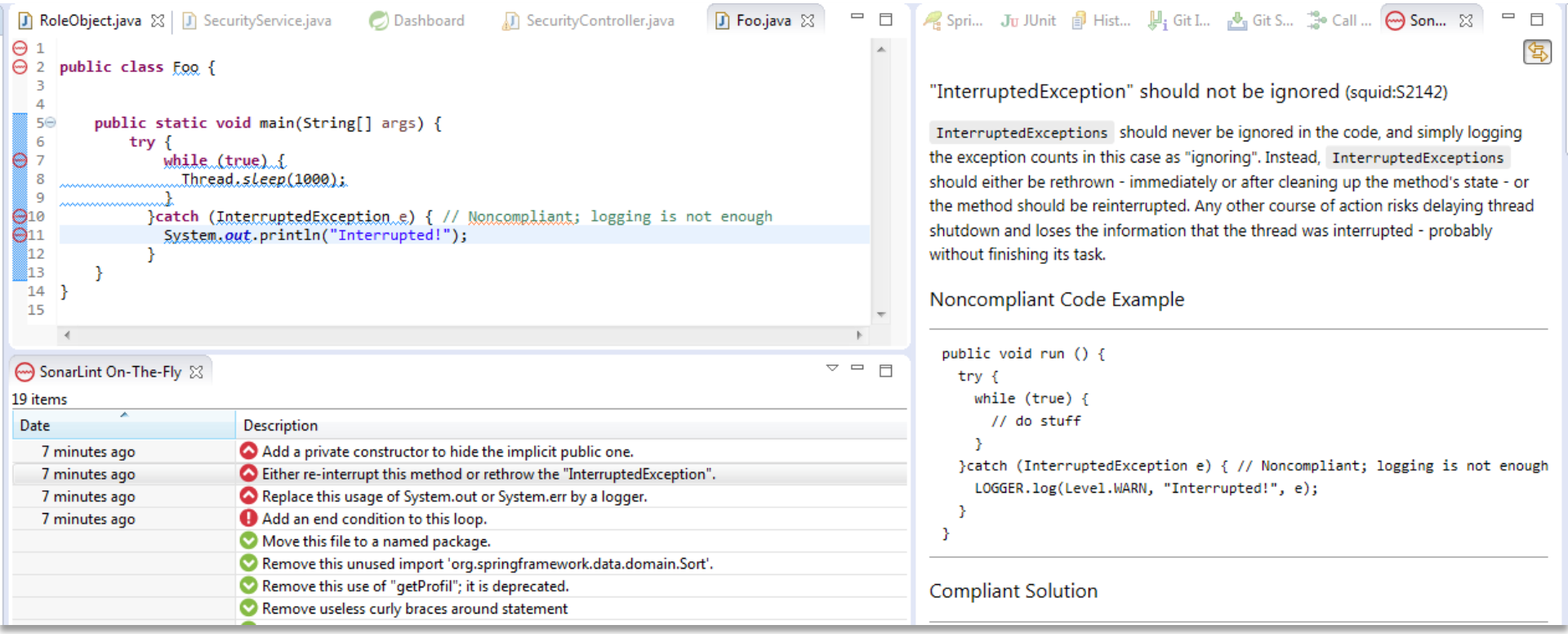
Des outils de forge

- Cela ne demande pas beaucoup d'effort et apporte **énormément**
- Un gestionnaire de version : **obligatoire**
  - Git (local) 
  - Gitlab (service) 
- Un service d'intégration continue : **fortement recommandé**
  - Jenkins, GitlabCI 
- Un outil d'analyse qualité : **recommandé**
  - Sonar 
- Un répertoire/proxy d'artefacts : *facultatif*
  - Nexus 

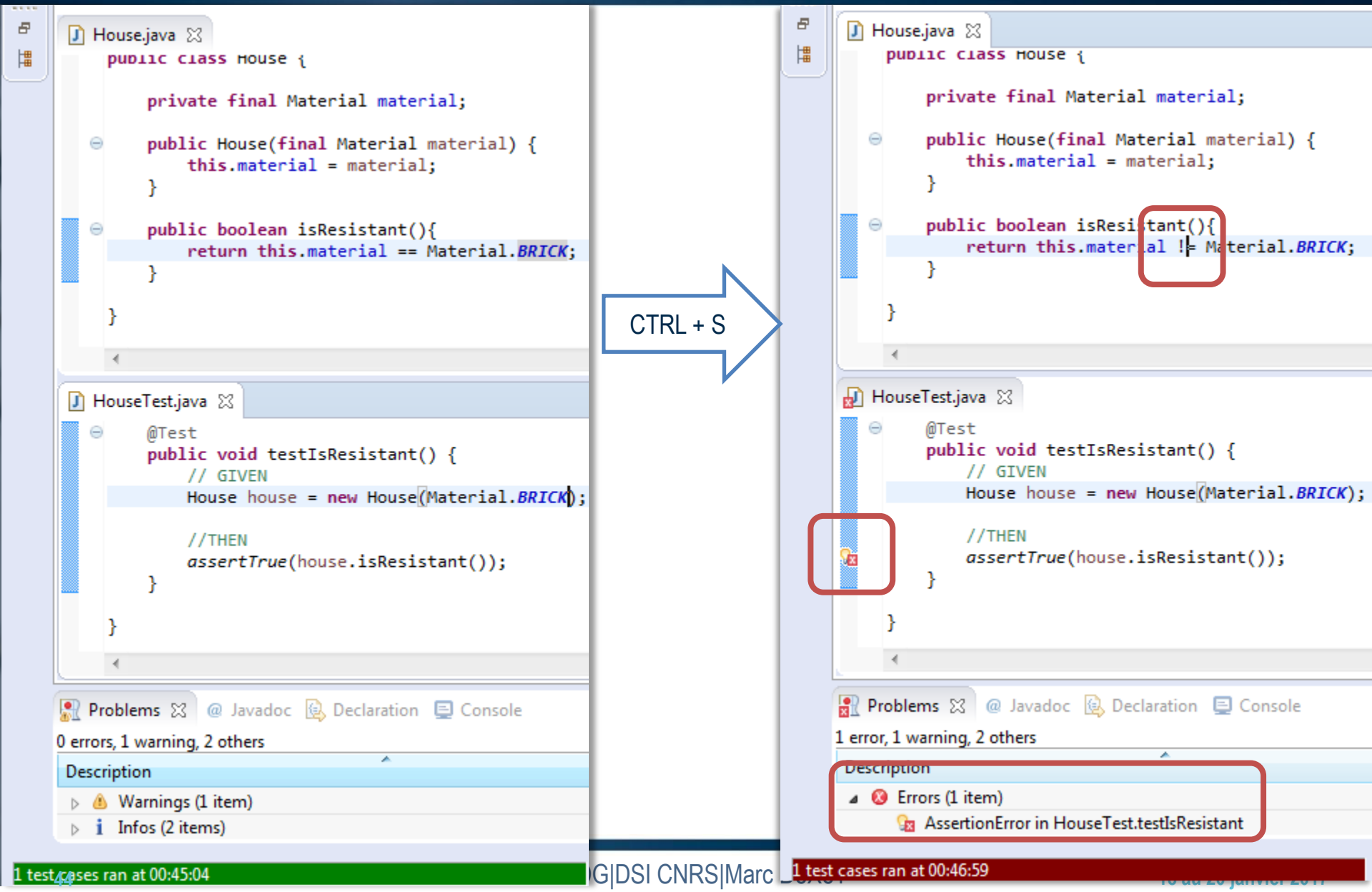
- Attention aux services exposés: prise en compte la sécurité ☺

Dans l'IDE
Sonar Lint

- Analyse du code en temps réel dans l'éditeur avec **SonarLint**
- Les règles sont liées à celle du serveur **Sonarqube**



Dans l'IDE Infinitest



www.cnrs.fr

Code

Analyse et revue

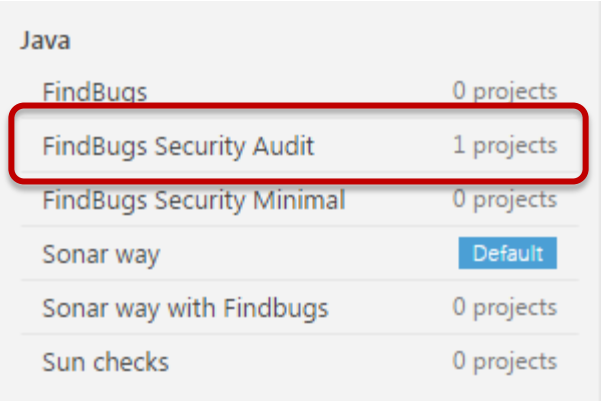
Reconsidérer

www.cnrs.fr

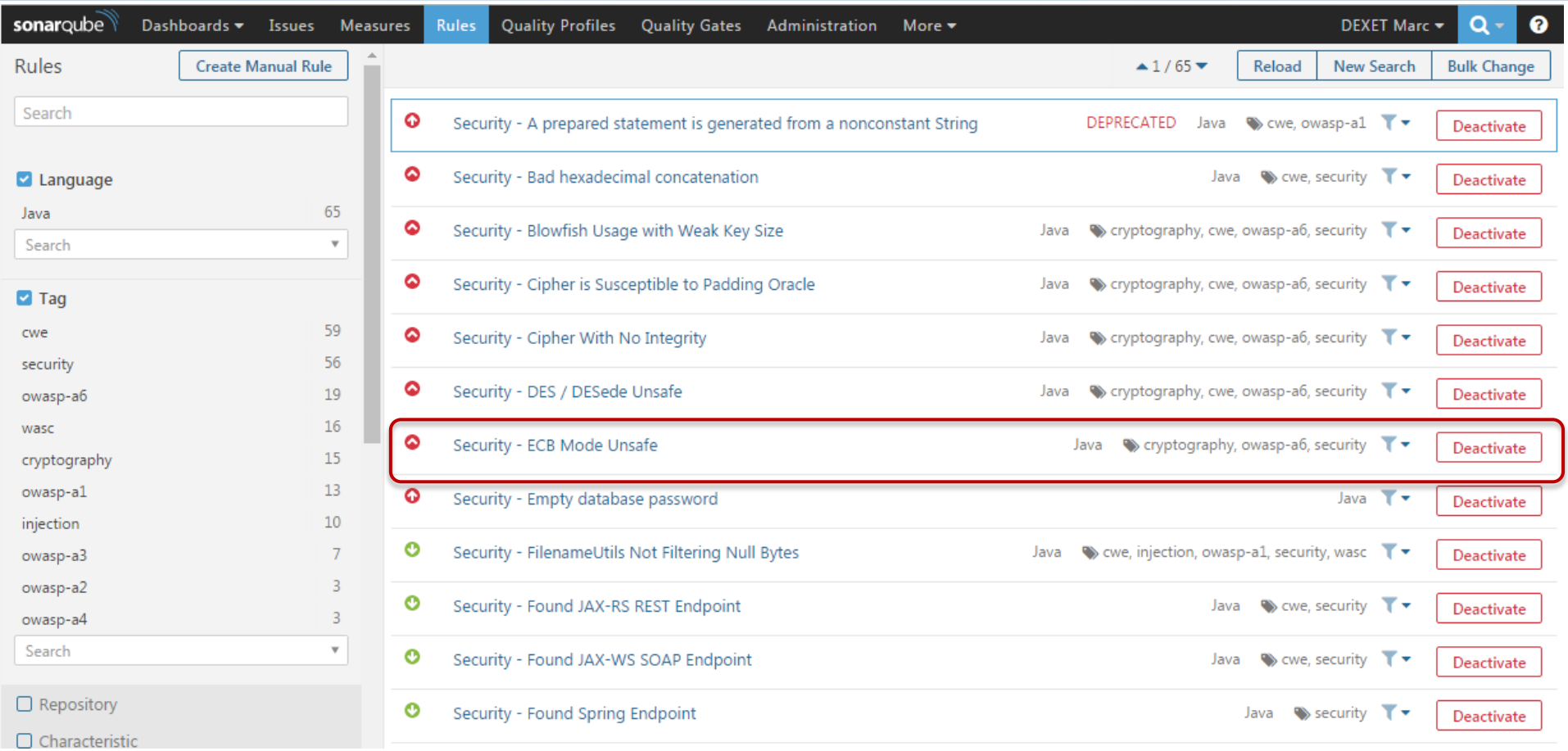
- Regarder le code sous un autre angle pour démasquer les vulnérabilités
  - Réussir à changer de perspective
  - Organiser une revue de code
- Plan
  - Changer de perspective

Analyse automatique

- Fonctionne sur des jeux de règles
  - Règles abusives ou non pertinentes : désactiver par projet
  - Faux positifs: un travail de filtrage est nécessaire
- Les solutions payantes peuvent être très onéreuses
- Exemple avec Sonar et java
  - Jeux de règles



Règles de sécurité (Sonar)



Security - ECB Mode Unsafe

findsecbugs:ECB_MODE

Major cryptography, owasp-a6, security Available Since 31 août 2015 Find Security Bugs (Java)

An authentication cipher mode which provides better confidentiality of the encrypted data should be used instead of Electronic Codebook (ECB) mode, which does not provide good confidentiality. Specifically, ECB mode produces the same output for the same input each time. So, for example, if a user is sending a password, the encrypted value is the same each time. This allows an attacker to intercept and replay the data.

To fix this, something like Galois/Counter Mode (GCM) should be used instead.

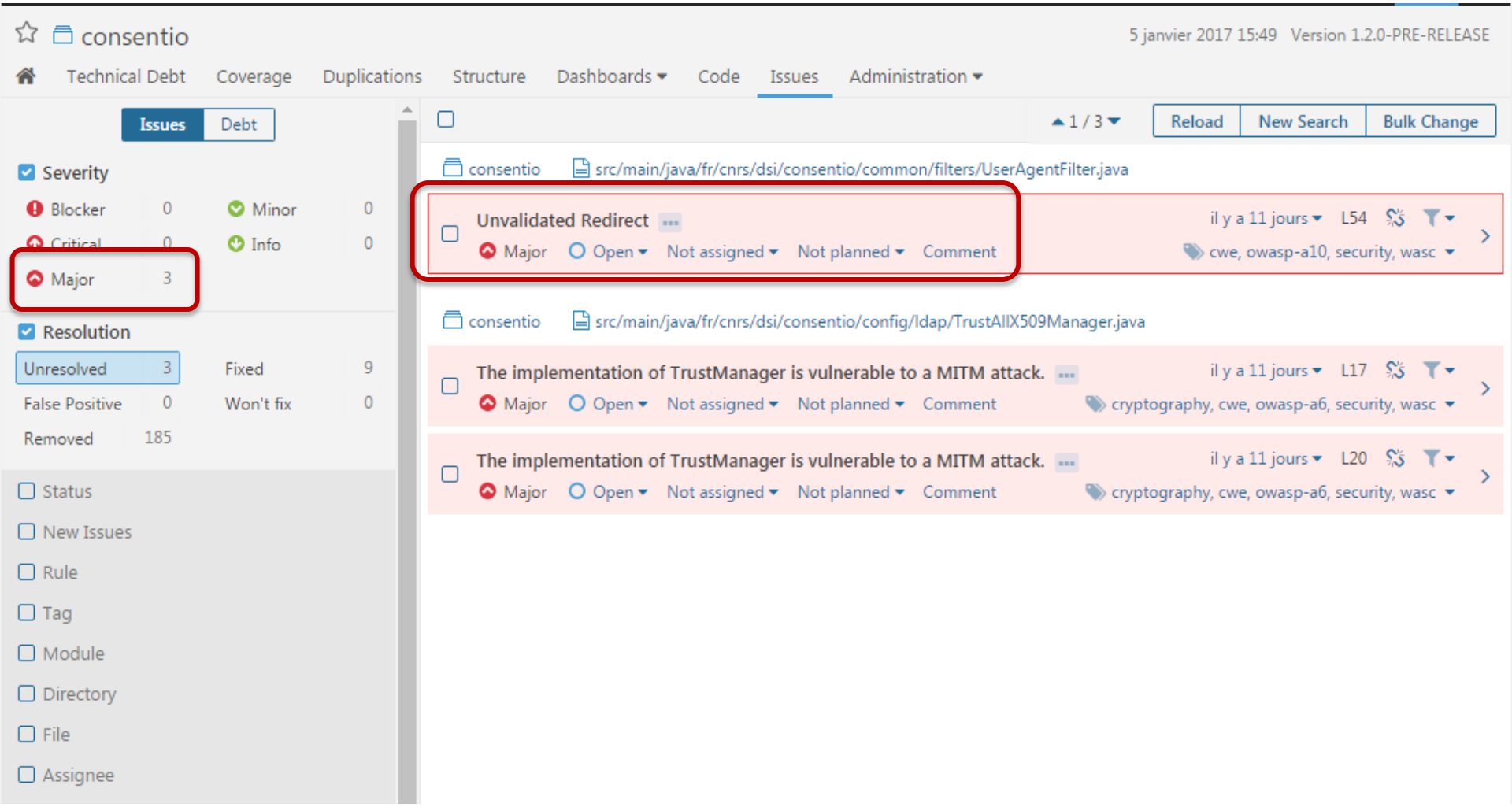
Code at risk:

```
Cipher c = Cipher.getInstance("AES/ECB/NoPadding");
c.init(Cipher.ENCRYPT_MODE, k, iv);
byte[] cipherText = c.doFinal(plainText);
```

Solution:

```
Cipher c = Cipher.getInstance("AES/GCM/NoPadding");
c.init(Cipher.ENCRYPT_MODE, k, iv);
byte[] cipherText = c.doFinal(plainText);
```

Exemple sur un projet réel



Visualisation du code

```
46          @Override
47          protected void doFilterInternal(HttpServletRequest request,
48                          HttpServletResponse response, FilterChain filterChain)
49                          throws ServletException, IOException {
50                  String userAgent = request.getHeader("user-agent");
51                  String requestURI = request.getRequestURI();
52
53                  if (requestURI.startsWith(staticContentPrefix) && requestURI.endsWith(".html") && ! requestURI.equalsIgnoreCase(redi
54                          response.sendRedirect(redirectLocation);
```

Unvalidated Redirect ...

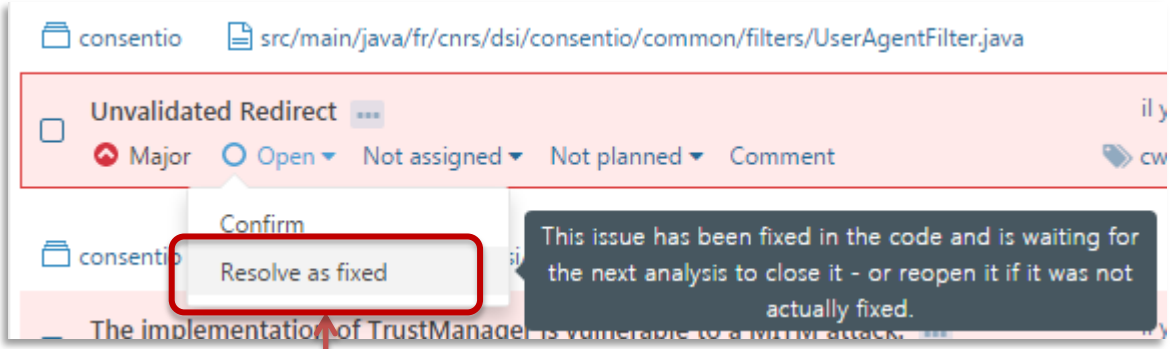
il y a 11 jours ▾ L54

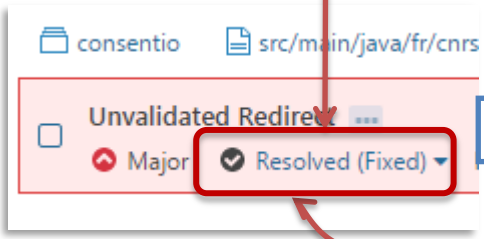
Major | Open ▾ | Not assigned ▾ | Not planned ▾ | Comment

cwe, owasp-a10, security, wa

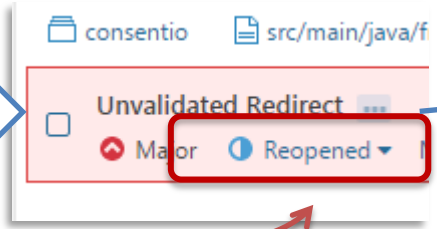
```
55                  } else {
56                          filterChain.doFilter(request, response);
57                  }
```

Résolution et faux positifs

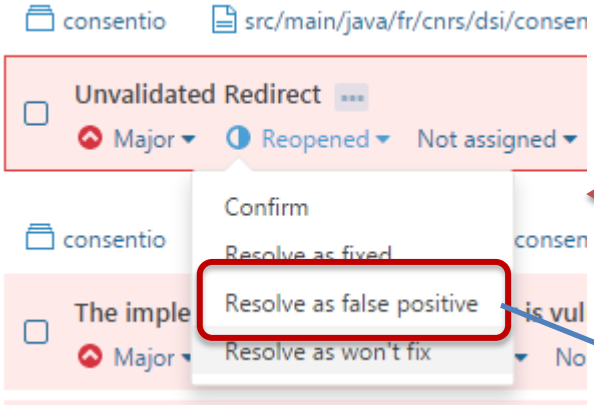


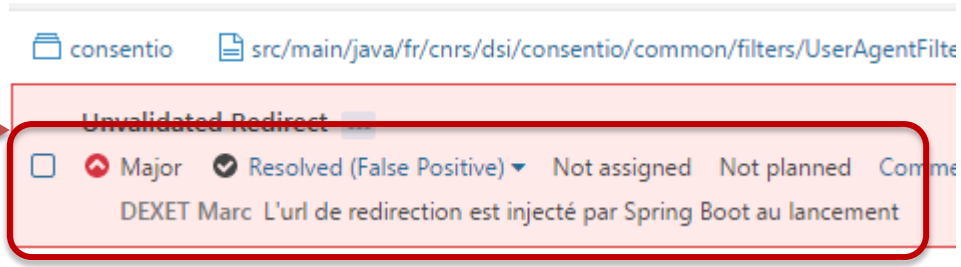


Nouvelle analyse

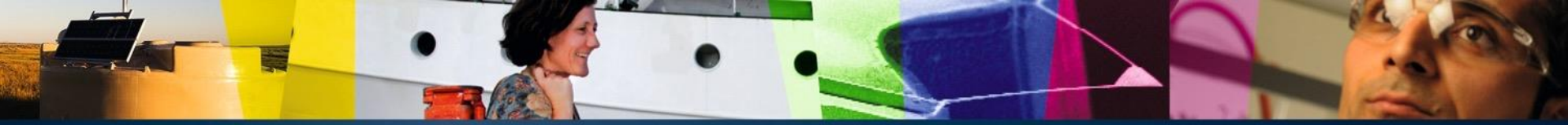


L'absence d'évolution est relevée





Marquage comme faux positif



Revue de code

- La revue de code est étonnamment efficace !
- La revue de code est un processus **technique** et **social**
  - augmentation de la qualité du code
  - appropriation *collective* du code,
    - compréhension du code
    - vision commune de la qualité et de la **sécurité**
  - montée en compétence *de tous les* membres d'une équipe
- Plusieurs approches possibles, consulter la littérature pour faire votre choix
  - "par-dessus l'épaule", de pair à pair, par pull-request, planifié…
- Mais il y a des règles pour que cela se passe bien… ou mal

La fameuse faille SSL Apple

Mauvaise qualité de code
Une revue de code aurait levé le problème

```
static OSStatus

SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,

uint8_t *signature, UInt16 signatureLen)

OSStatus err;

...

if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)

goto fail;

if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)

goto fail;

goto fail;

if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)

goto fail;

...

fail:

SSLFreeBuffer(&signedHashes);

SSLFreeBuffer(&hashCtx);

return err;

}
```

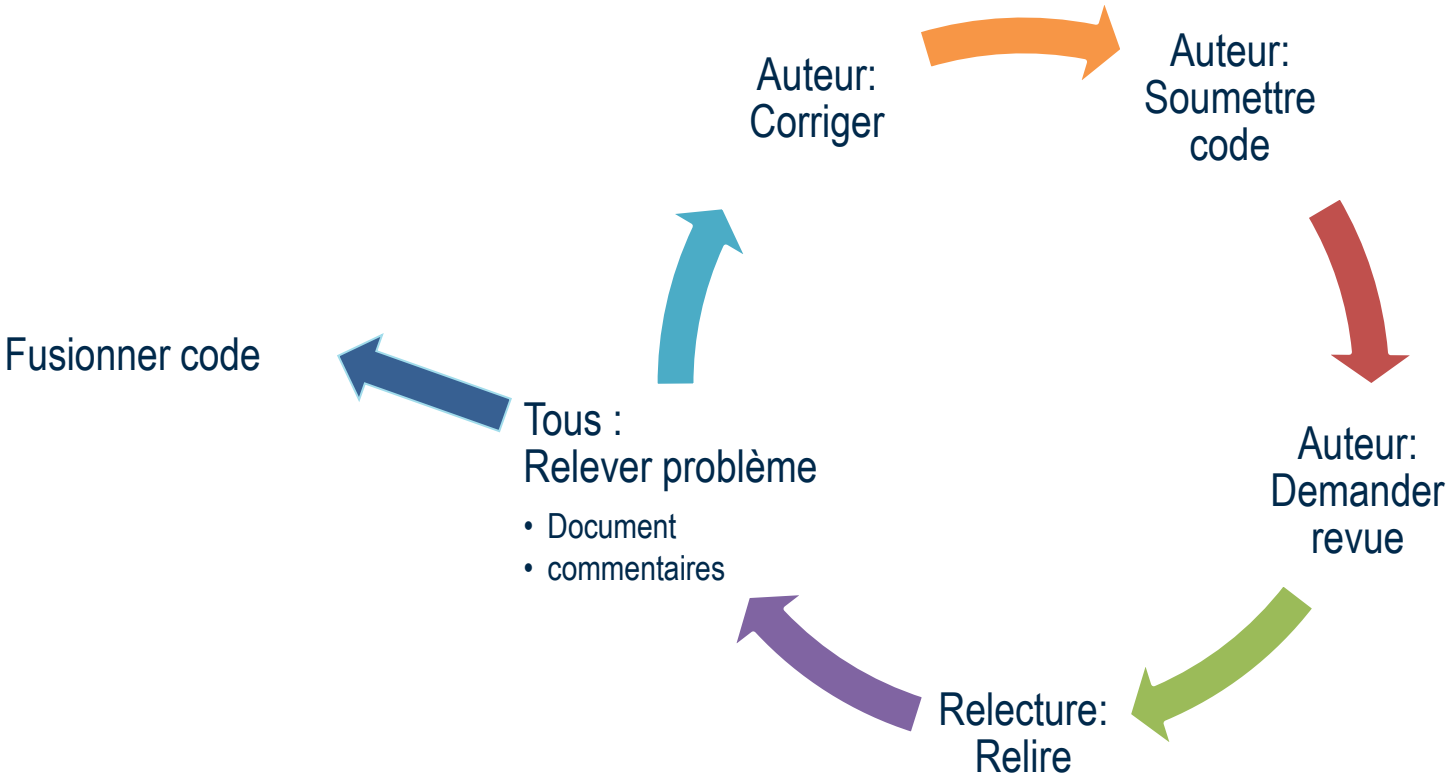
Principes de la revue de code bienveillante

www.cnrs.fr

- Pour être efficace, la revue doit être **bienveillante** (et non redoutée)
- Travailler sur son **égo**, cf. "*Egoless Programming*"
  - Vous faites des erreurs,
  - **Vous n'êtes pas votre code**,
  - La seule autorité provient de la connaissance, pas de la position
  - Soyez sympa avec le développeur, dur avec le code
  - …
- Etre dans un esprit constructif et se plier au jeu, expert comme novice
- Apporter de la valeur ajoutée
  - Laisser les outils automatiques faire leur travail
  - Chercher les problèmes d'architecture, de pattern
- Commencer par des petits bouts (commit), parties sensibles ou complexes

https://blog.codinghorror.com/the-ten-commandments-of-egoless-programming/

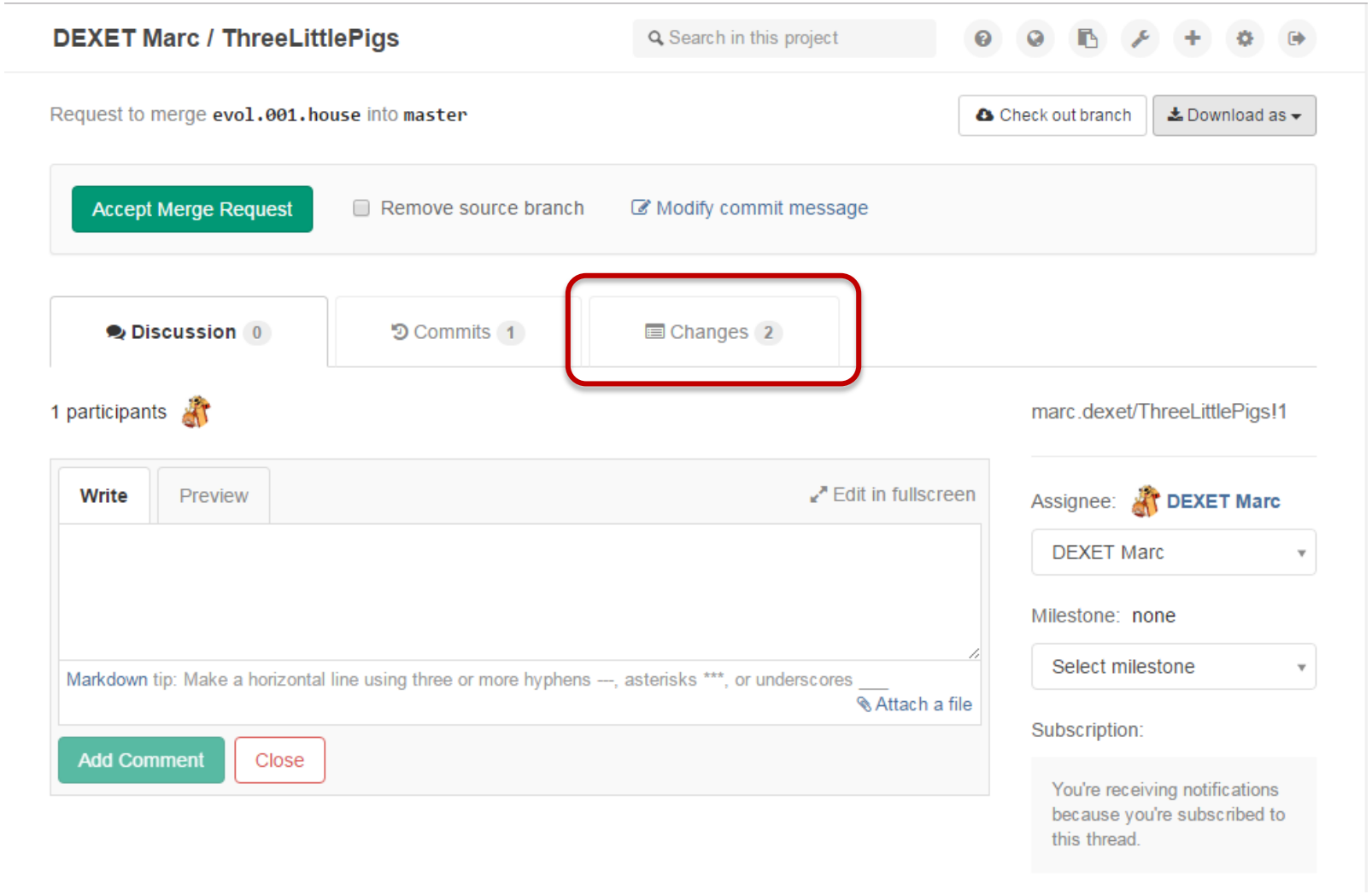
Cycle de revue de code



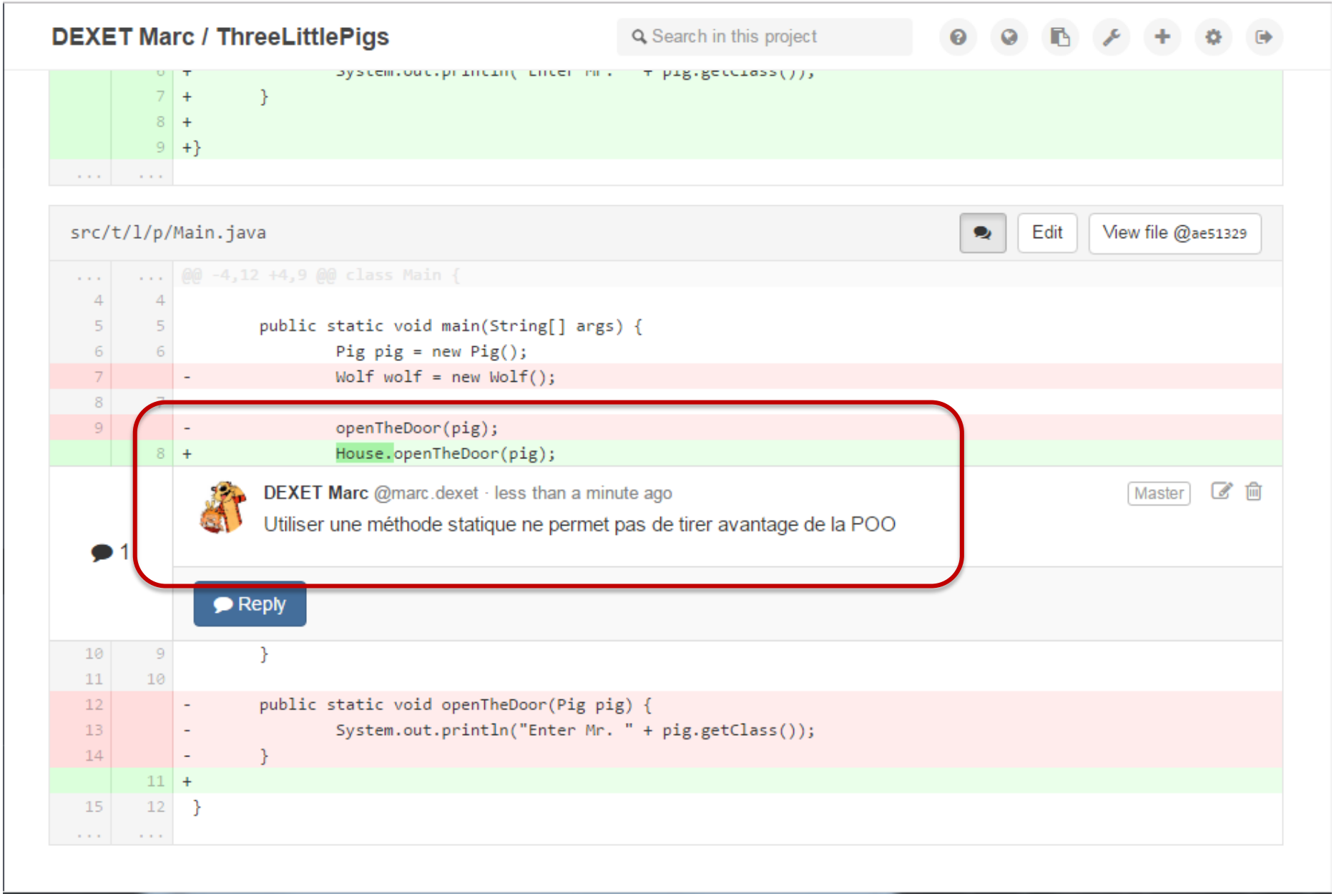
Comment faire ?

- Préparer un minimum la revue avant
  - Lecture de la portion de code
  - Ne pas corriger en séance, c'est l'auteur qui corrige
- 1 minute par défaut, au-delà, c'est qu'il y a débat
- Se limiter dans le temps et le volume
  - *"Ask programmers to review 10 lines of code, they'll find 10 issues. Ask them to do 500 lines and they'll say it looks good".*
- **Documenter** les *valeurs* et *conventions* de code,
  - si des manquements ou des désaccords sont relevés lors des revues,
  - si elles n'existent pas encore.
- Garder une trace des points relevés et des corrections apportées
- Trouver sa voie
  - http://blog.octo.com/revue-de-code-quel-format-choisir/

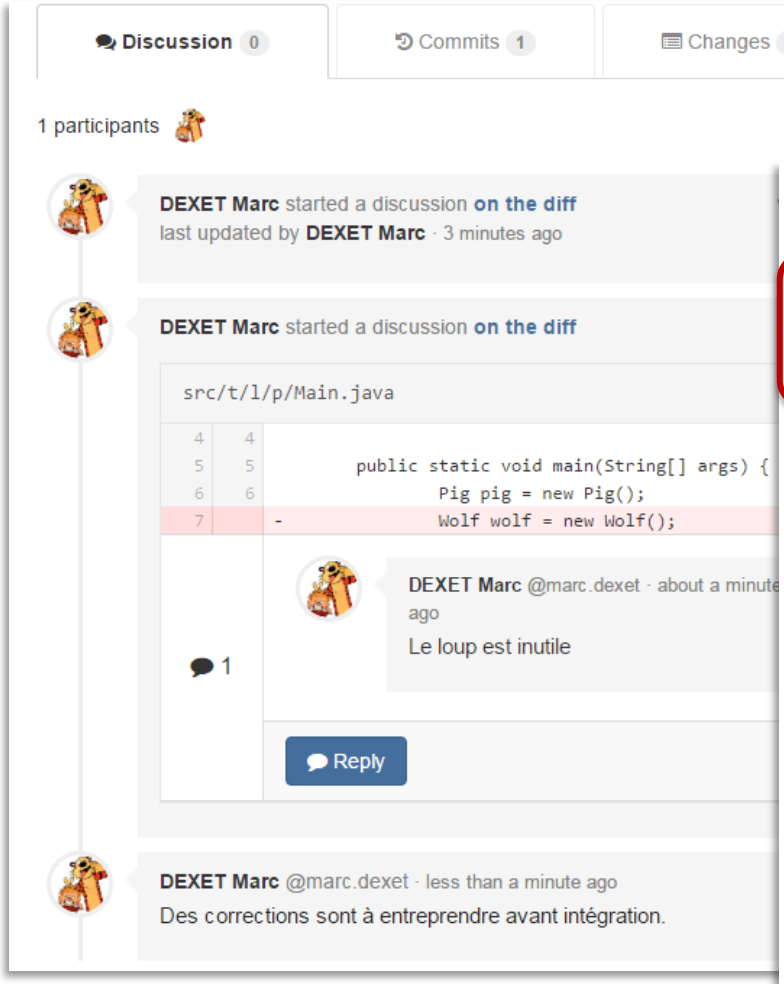
Revue sur merge request (Gitlab)

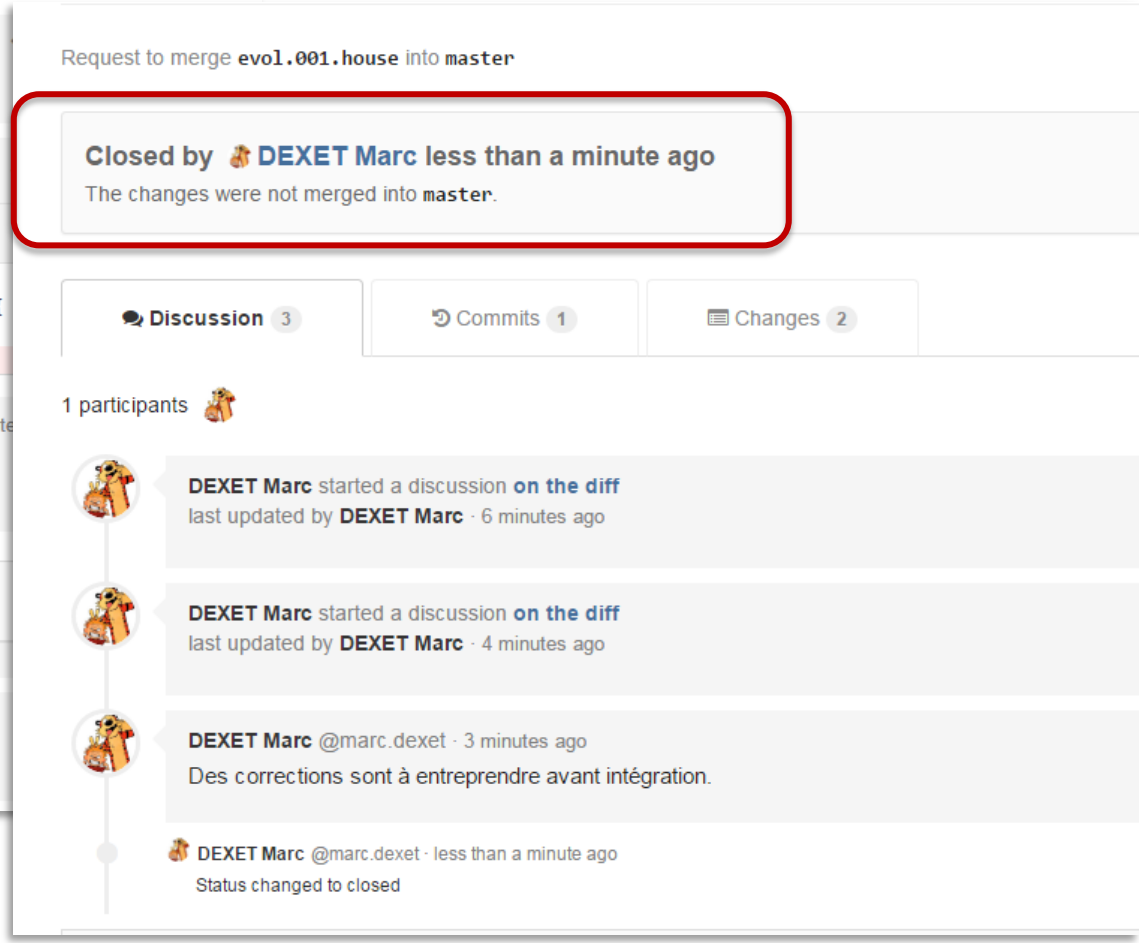


Le relecteur signale des problèmes

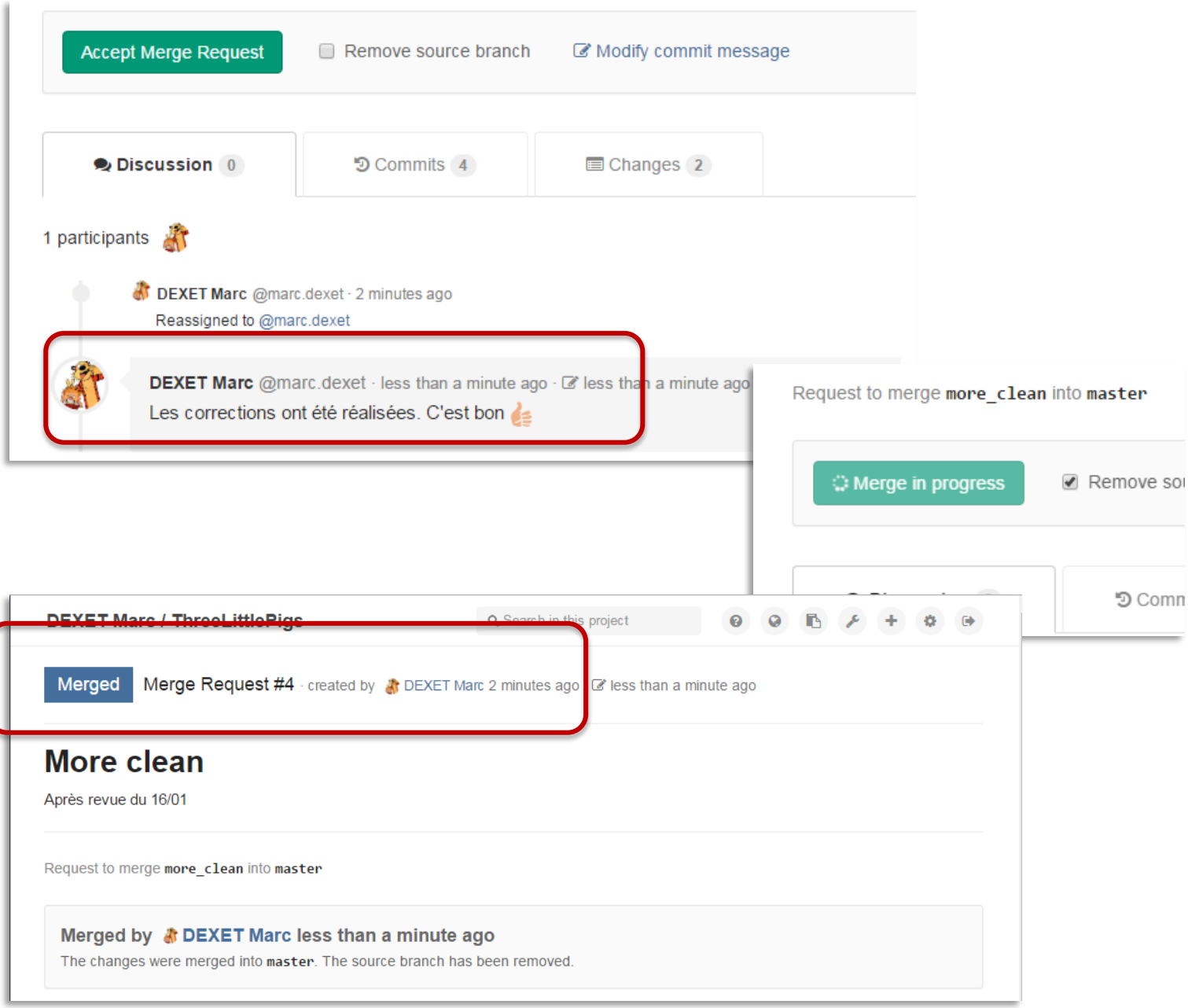


La décision prise est de ne pas fusionner

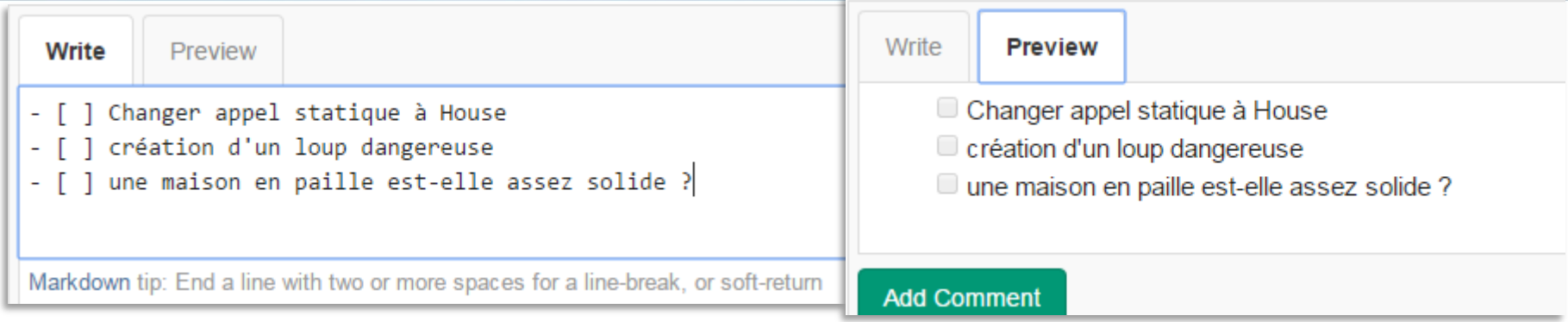


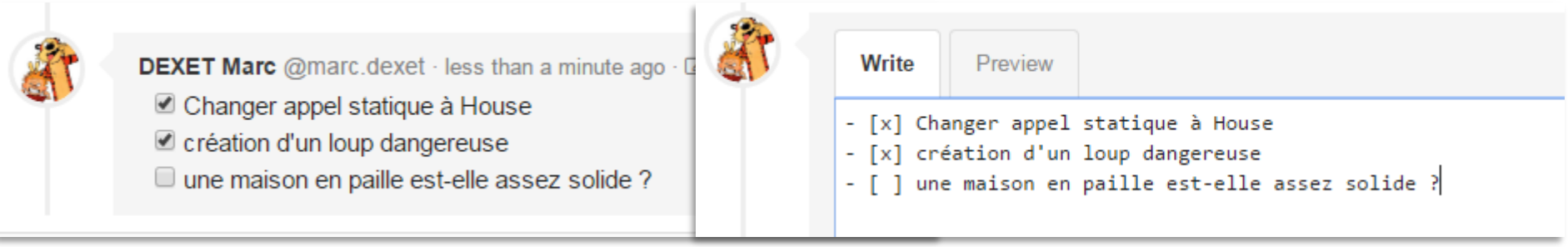


L'auteur soumet une nouvelle version, la fusion est réalisée



Faites des listes





Usage de check-list

- Il existe des check-lists "modèles"
  - http://software-security.sans.org/resources/swat
  - https://www.owasp.org/images/0/08/OWASP_SCP_Quick_Reference_Guide_v2.pdf
- Sélectionnez en une, adaptez la à votre réalité
- Intégrez la dans votre cycle de développement
  - Avant livraison
  - A la fin d'un Sprint
  - A chaque feature…
- Pensez à votre dernière révision automobile ☺

DATA PROTECTION

BEST PRACTICE	DESCRIPTION	CWE ID
Use HTTPS Everywhere	Ideally, HTTPS should be used for your entire application. If you have to limit where it's used, then HTTPS must be applied to any authentication pages as well as to all pages after the user is authenticated. If sensitive information (e.g. personal information) can be submitted before authentication, those features must also be sent over. **EXAMPLE**: Firesheep	CWE-311 CWE-319 CWE-523
Disable HTTP Access for All Protected Resources	For all pages requiring protection by HTTPS, the same URL should not be accessible via the insecure HTTP channel.	CWE-319
Use the Strict-Transport-Security Header	The Strict-Transport-Security header ensures that the browser does not talk to the server over HTTP. This helps reduce the risk of HTTP downgrade attacks as implemented by the sslsniff tool.	
Store User Passwords Using a Strong, Iterative, Salted Hash	User passwords must be stored using secure hashing techniques with strong algorithms like PBKDF2, bcrypt, or SHA-512. Simply hashing the password a single time does not sufficiently protect the password. Use adaptive hashing (a work factor), combined with a randomly generated salt for each user to make the hash strong. **EXAMPLE**: LinkedIn password leak	CWE-257
Securely Exchange Encryption Keys	If encryption keys are exchanged or pre-set in your application then any key establishment or exchange must be performed over a secure channel	
Set Up Secure Key Management Processes	When keys are stored in your system they must be properly secured and only accessible to the appropriate staff on a need to know basis. **EXAMPLE**: AWS Key Management Service (KMS), Azure Key Vault, AWS CloudHSM	CWE-320
Weak TLS Configuration on Servers	Weak ciphers must be disabled on all servers. For example, SSL v2, SSL v3, and TLS protocols prior to 1.2 have known weaknesses and are not considered secure. Additionally, disable the NULL, RC4, DES, and MD5 cipher suites. Ensure all key lengths are greater than 128 bits, use secure renegotiation, and disable compression. **EXAMPLE**: Qualys SSL Labs	
Use Valid HTTPS Certificates from a Reputable CA	HTTPS certificates should be signed by a reputable certificate authority. The name on the certificate should match the FQDN of the website. The certificate itself should be valid and not expired. **EXAMPLE**: Let's Encrypt (https://letsencrypt.org)	
Disable Data Caching Using Cache Control Headers and Autocomplete	Browser data caching should be disabled using the cache control HTTP headers or meta tags within the HTML page. Additionally, sensitive input fields, such as the login form, should have the autocomplete=off setting in the HTML form to instruct the browser not to cache the credentials.	CWE-524
Limit the Use and Storage of Sensitive Data	Conduct an evaluation to ensure that sensitive data is not being unnecessarily transported or stored. Where possible, use tokenization to reduce data exposure risks.	

Application Security Verification Standard Project (ASVSP)

www.cnrs.fr

- Application Security Verification Standard Project (ASVSP)
  - Un site OWASP Application Security Verification Standard Project
  - Un guide PDF de version 3.0.1 (anglais)
  - Un outil OWASP Security Knowledge Framework en relation (pas testé)

- Contrairement au top 10 OWASP qui décrit ce qu'il ne faut pas faire, AVSP est proactif et constitue un plan d'action.

Domaines couverts (ASVSP)

- Architecture, design et threat modelling
- Authentification
- Gestion des sessions
- Contrôle des accès
- Traitement des entrées malveillantes
- Cryptographie générale (*at rest* vs *in transit*)
- Traitement des erreurs et historisation
- Protection des données (confidentialité, intégrité, disponibilité)
- Communications (TLS, cryptographie)
- Configuration sécurité HTTP
- Malveillance
- bombes, backdoor, comportement
- Logique métier
- Fichiers et ressources suspicieux
- Application Mobile
- Web services
- Configuration

Niveaux (ASVSP)

- Chaque domaine comprend
  - une explication des objectifs
  - des exigences modulées par niveau
  - des pointeurs vers des références

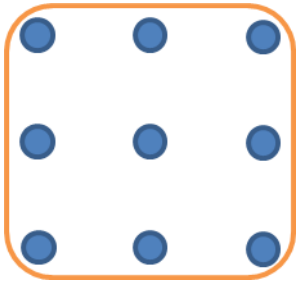
- Niveaux
  - niveau 1 *opportuniste*
    - applicable à toutes les applications
  - niveau 2 *standard*
    - à appliquer pour les applications à données **sensibles** nécessitant une protection
  - niveau 3 *avancé*
    - adapté aux applications **critiques**, avec des données **très sensibles**, à **forte valeur** ou avec un **besoin de confiance élevé**

ASVSP

#	Description	1	2	3	Since
11.1	Verify that the application accepts only a defined set of required HTTP request methods, such as GET and POST are accepted, and unused methods (e.g. TRACE, PUT, and DELETE) are explicitly blocked.	✓	✓	✓	1.0
11.2	Verify that every HTTP response contains a content type header specifying a safe character set (e.g., UTF-8, ISO 8859-1).	✓	✓	✓	1.0
11.3	Verify that HTTP headers added by a trusted proxy or SSO devices, such as a bearer token, are authenticated by the application.		✓	✓	2.0
11.4	Verify that a suitable X-FRAME-OPTIONS header is in use for sites where content should not be viewed in a 3rd-party X-Frame.		✓	✓	3.0.1
11.5	Verify that the HTTP headers or any part of the HTTP response do not expose detailed version information of system components.	✓	✓	✓	2.0
11.6	Verify that all API responses contain X-Content-Type-Options: nosniff and Content-Disposition: attachment; filename="api.json" (or other appropriate filename for the content type).	✓	✓	✓	3.0
11.7	Verify that a content security policy (CSPv2) is in place that helps mitigate common DOM, XSS, JSON, and JavaScript injection vulnerabilities.	✓	✓	✓	3.0.1
11.8	Verify that the X-XSS-Protection: 1; mode=block header is in place to enable browser reflected XSS filters.	✓	✓	✓	3.0

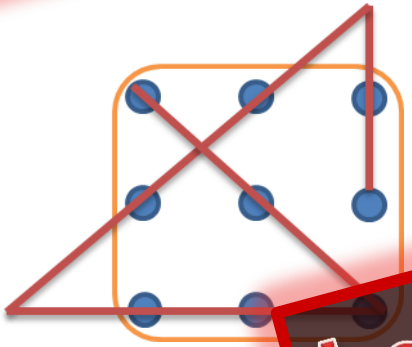
www.cnrs.fr

Culture Sécurité



Exigence
Le système ne doit pas permettre de relier les 9 points avec 4 traits droits, sans discontinuité.

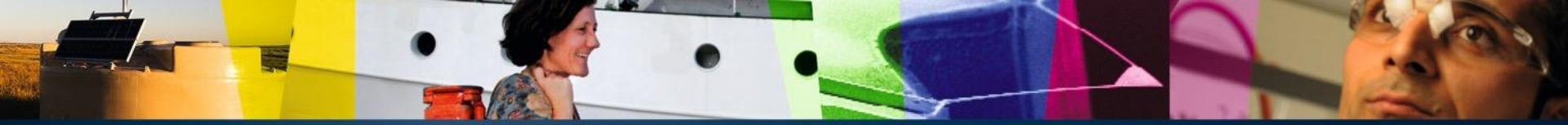
Développeur
Le système est valide, la preuve...



L'attaquant
Oh wait…

Assumption is mother of all screw up

Les hypothèses sont à la base de tous les foirages



Penser autrement, sortir du cadre

www.cnrs.fr

- Sortir du cadre
  - Le développeur pense à ce que son code **doit faire**
  - L'attaquant à ce que le code **ne devrait pas laisser faire**
- C'est une attitude à prendre
  - ➔ Attitude vigilante du motard
    - Ne pas penser à ce qui **doit** se passer (la voiture s'arrête au stop)
    - Mais à ce qui **peut** se passer (la voiture peut griller le stop)
- Etat d'esprit des tests unitaires
  - Pour 1 cas nominal, ***N*** cas anormaux

Faire connaître les bonnes pratiques
(psychologie appliquée)

www.cnrs.fr

- Faire connaître le top 10 en français
  - L'imprimer, le laisser trainer en salle café, couvrir le mur de la café' ☺
- Faire des réunions de sensibilisation
  - Brown-Bag Sessions, …
- Gamification, ludification, Serious Game…
  - Applications vulnérables
    - http://www.dvwa.co.uk/
    - OWASP WebGoat
  - http://www.gameofhacks.com/
  - https://xss-game.appspot.com
  - Jeux de cartes
    - https://www.owasp.org/index.php/OWASP_Cornucopia

- Design patterns
  - https://fr.wikibooks.org/wiki/Patrons_de_conception
  - https://en.wikipedia.org/wiki/Software_design_pattern
  - http://www.w3sdesign.com/
  - Un cours en français https://dpt-info.u-strasbg.fr/~blansche/files/ogl_cours_5.pdf
- Anti-patterns
  - https://fr.wikipedia.org/wiki/Antipattern#Anti-patrons_de_d.C3.A9veloppement
  - très drôle et grinçant https://fr.wikipedia.org/wiki/Le_Mythe_du_mois-homme
- Liste de compétences
  - Vous pouvez vous inspirer de http://sijinjoseph.com/programmer-competency-matrix/
  - ou la version sous forme de questionnaire http://competency-checklist.appspot.com/