RRABIDS: un système de détection d'intrusions comportemental destiné aux applications Web écrites en Ruby on Rails.

Romaric Ludinard, Loïc Le Hennaf, Éric Totel

Supélec

Vendredi 10 juin 2011

Plan de la présentation

Projet Dali

Design and Assessment of application Level Intrusion detection systems

- Conception et évaluation d'IDS au niveau applicatif.
- Partenaires :
  - Kereval : PME rennaise spécialisée dans le test applicatif.
  - Télécom-Bretagne.
  - Supélec.
  - Laas-Cnrs.

Détection d'intrusions

Par signature :

- Modélise le comportement interdit de l'entité
- Utilise une base de connaissance des attaques connues
- Ne permet pas de découvrir de nouvelles attaques

Comportementale :

- Modélise le comportement normal de l'entité
- Repose sur la capacité à créer un modèle de référence complet et précis
- Permet de détecter de nouvelles attaques sans intervention sur l'IDS ni mise à jour

IDS applicatif

Type d'erreurs détectables

- Erreur dans le flot d'exécution :
  - Les instructions, appels de fonction, retour de fonction ne s'exécutent pas dans l'ordre correct.
  - Témoigne d'attaques très connues et très répandues telles que les dépassement de buffer (*buffer overflow*), détournement vers les librairies (*return to lib C*, *return-oriented programming*).
  - Déjà largement étudié.
- Erreur dans les données :
  - Certaines variables prennent des valeurs incorrectes vis à vis de la spécification du logiciel (pas de son implémentation évidemment).
  - Classe d'attaque aussi dangereuse que les précédentes.
  - Par exemple débordement d'entiers (*integer overflow*).
  - Beaucoup moins étudié.

En résumé

Dans ces travaux nous nous intéressons à :

- Détection comportementale ;
- au niveau applicatif ;
- par détection d'attaques contre les données.

Pour cela :

- Construction des invariants du code à vérifier.
- La violation d'un invariant peut être le témoignage de l'exploitation d'une vulnérabilité de l'application par un attaquant.

Ruby

- Langage interprété développé depuis 1993 par Yukihiro Matsumoto.
- Langage orienté-objet.
- Typage dynamique.
- Gestion automatique de la mémoire (garbage-collector).
- Support de la réflexivité.

Mais :

- Langage sans spécification ni grammaire.
- L'implémentation originale sert de référence *de facto*.

Ruby on Rails

- Framework destiné à la création d'applications Web.
- Utilisation du modèle MVC (Modèle-Vue-Contrôleur).
- Prototypage rapide d'applications Web.
- Largement utilisé, par exemple par Twitter pour l'interface graphique de leur site Web.

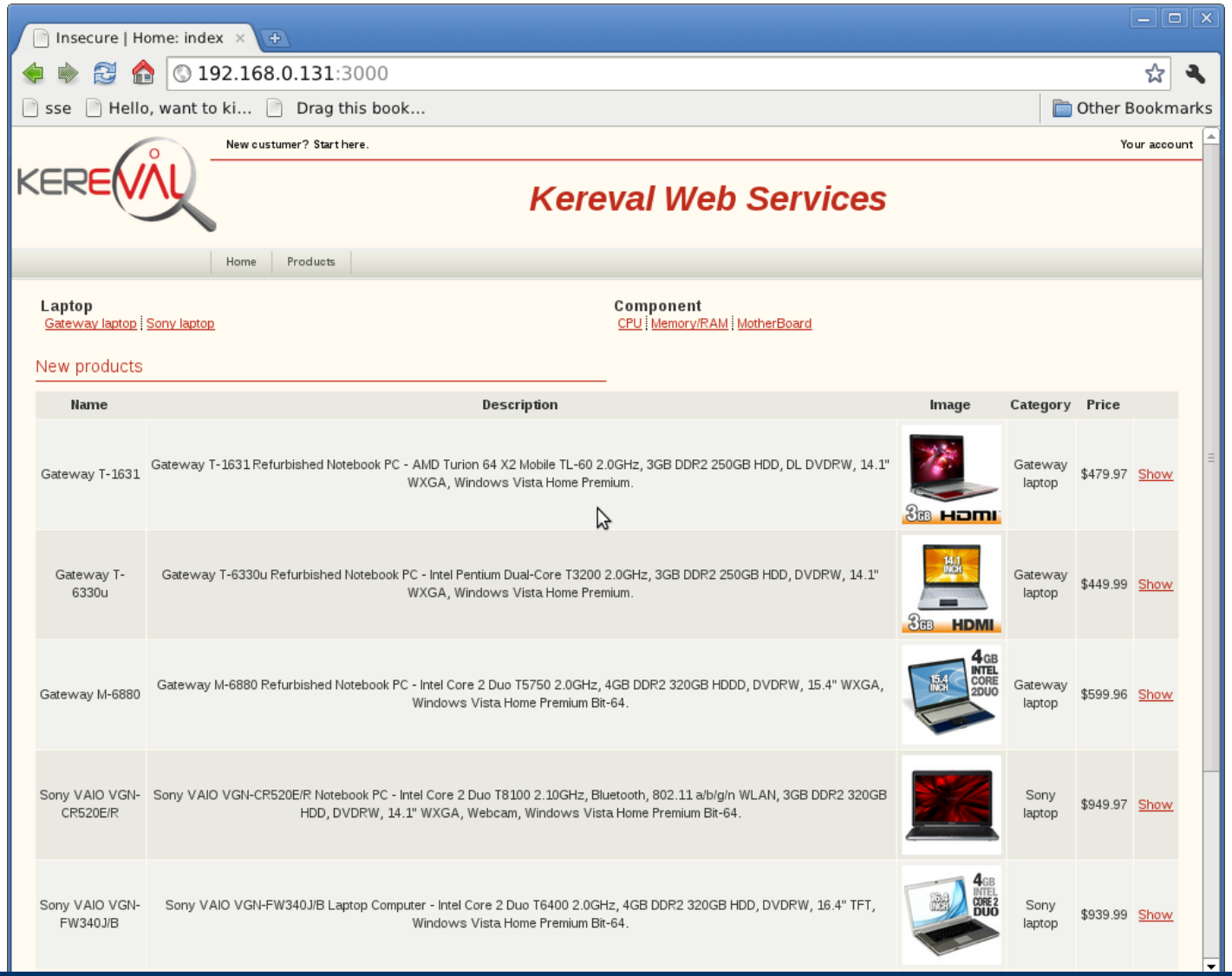
Modèle MVC

- Modèle : Le modèle défini les données de l'application et les méthodes d'accès. Tout les traitements sont effectués dans cette couche.
- Vue : La vue prend les informations en provenance du modèle et les présente à l'utilisateur.
- Controlleur : Le contrôleur répond aux événements de l'utilisateur et commande les actions sur le modèle. Cela peut entrainer une mise à jour de la vue.

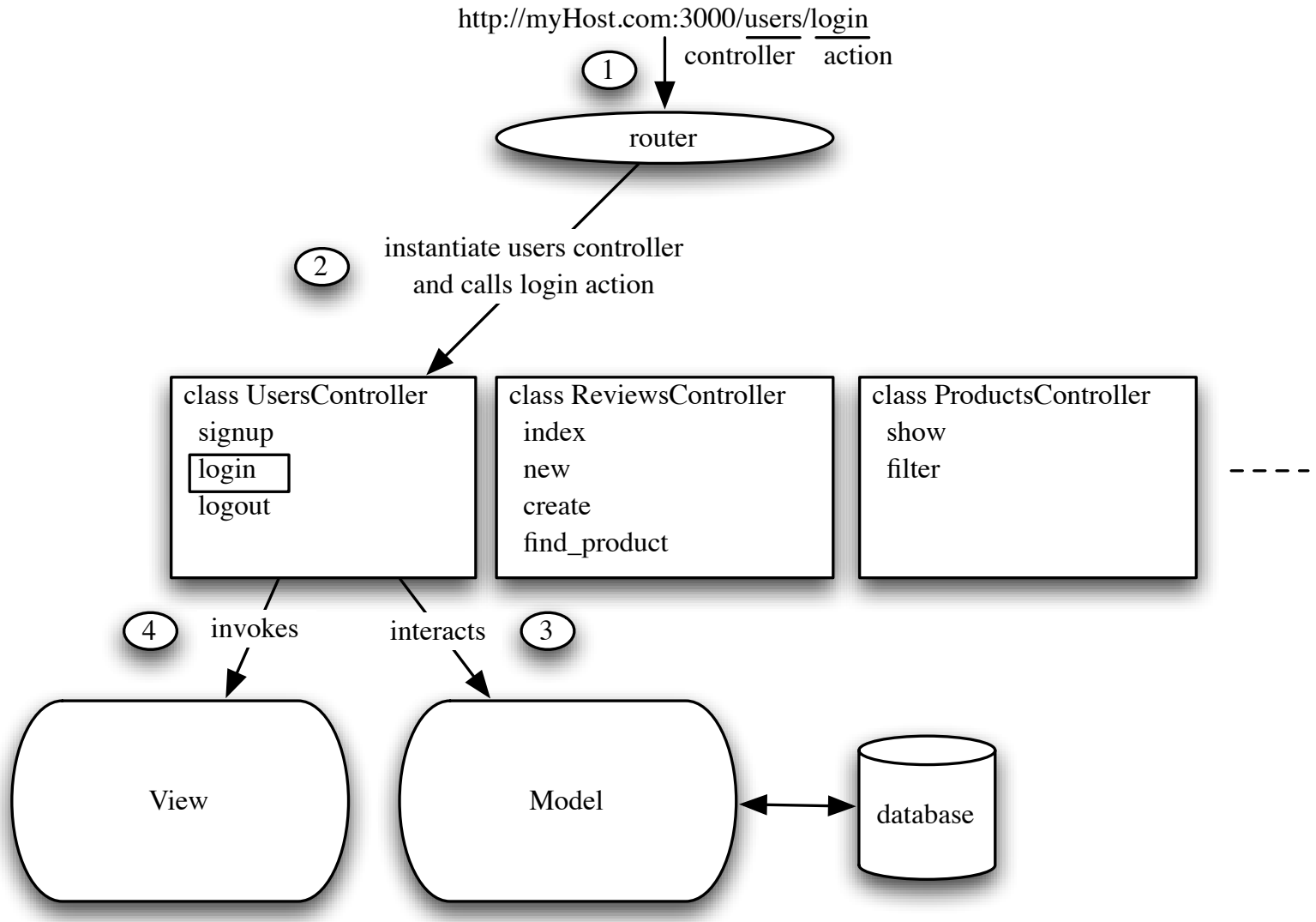
Application Insecure

- Application développée par Kereval.
- Développée en Ruby en utilisant le framework Ruby On Rails (RoR) destiné aux applications Web.
- Il s'agit d'une application de commerce en ligne.
- Application trois-tiers :
  - Serveur Web.
  - Interpréteur Ruby embarqué.
  - Base de données (MySQL).

Application Insecure



Application Insecure

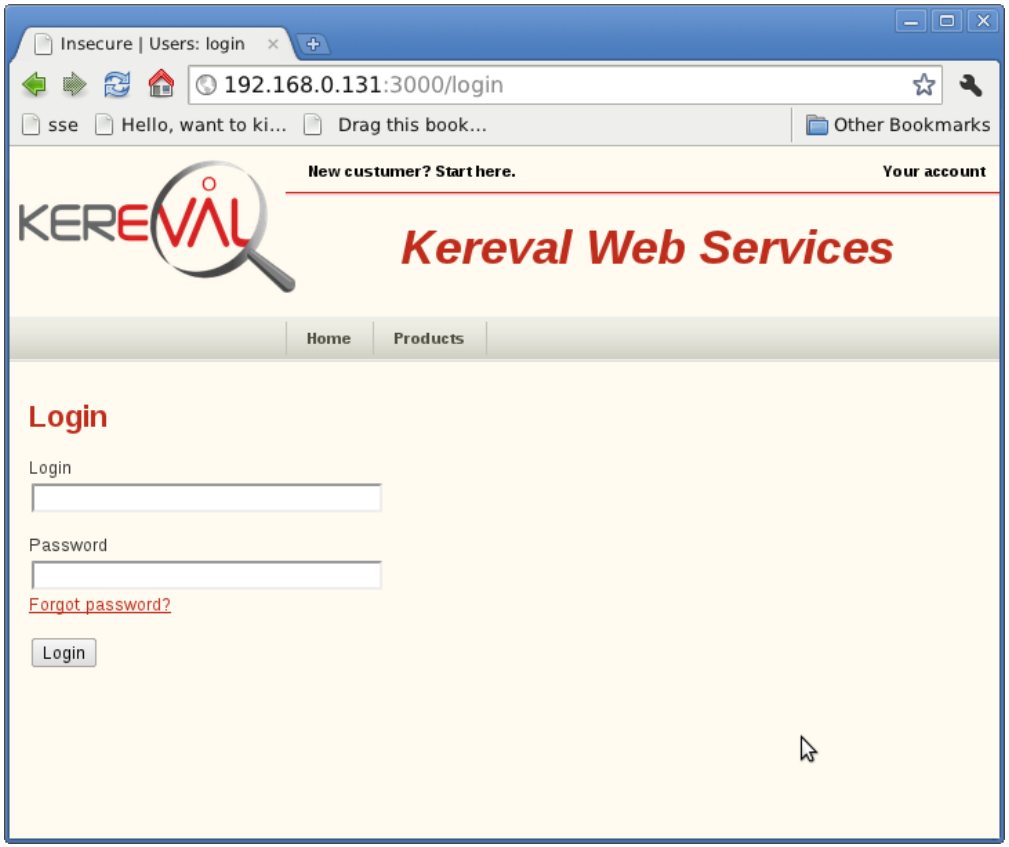


Failles dans l'application Insecure

Cette application présente certaines des failles communément rencontrées dans ce type d'applications :

- Injection SQL.
- Modification des paramètres des requêtes.
- Manipulation de cookie.
- Cross-site scripting.

Application Insecure



Exemple d'attaque

Injection SQL

- Saisie des informations *xxxx 'OR' '1'='1'* pour les champs *login* et *password*.
- La requête émise vers la base de données est *select * from table user where login=xxxx 'OR '1'='1' and password = xxxx 'OR '1'='1'*
- Permet de se connecter en tant qu'administrateur du site de vente sans en connaître le mot de passe.

Authentification sur l'application

```
class UsersController < ApplicationController
 def login
    if request.post?
       # whole user is stored in the session
       if session[:user] = User.authenticate(params[:user][:login],
params[:user][:password])
          flash[:notice]  = 'You have been successfully logged in.'
          if session[:user].admin
            redirect_to :controller => '/admin/home', :action => 'index'
          else
            redirect_to :controller => '/user/home',  :action => 'index'
          end
       else
          flash.now[:error] = 'Login failed'
       end
   end
```

params[:user][:login] == "admin"

params[:user][:password] == "secret"

Authentification sur l'application

```
class UsersController < ApplicationController
 def login
    if request.post?
       # whole user is stored in the session
       if session[:user] = User.authenticate(params[:user][:login],
params[:user][:password])
          flash[:notice]  = 'You have been successfully logged in.'
          if session[:user].admin
            redirect_to :controller => '/admin/home', :action => 'index'
          else
            redirect_to :controller => '/user/home',  :action => 'index'
          end
       else
          flash.now[:error] = 'Login failed'
       end
   end
```

params[:user][:login] == "admin"

params[:user][:password] == "secret"

session[:user][:login] == "admin"

Authentification sur l'application

```
class UsersController < ApplicationController
 def login
    if request.post?
       # whole user is stored in the session
       if session[:user] = User.authenticate(params[:user][:login],
params[:user][:password])
          flash[:notice]  = 'You have been successfully logged in.'
          if session[:user].admin
            redirect_to :controller => '/admin/home', :action => 'index'
          else
            redirect_to :controller => '/user/home',  :action => 'index'
          end
       else
          flash.now[:error] = 'Login failed'
       end
   end
```

params[:user][:login] == "admin"

params[:user][:password] == "secret"

session[:user][:login] == "admin"

params[:user][:login] == session[:user][:login]

Impact de l'attaque au niveau de l'application

```
class UsersController < ApplicationController
 def login
    if request.post?
       # whole user is stored in the session
       if session[:user] = User.authenticate(params[:user][:login],
params[:user][:password])
          flash[:notice]  = 'You have been successfully logged in.'
          if session[:user].admin
            redirect_to :controller => '/admin/home', :action => 'index'
          else
            redirect_to :controller => '/user/home',  :action => 'index'
          end
       else
          flash.now[:error] = 'Login failed'
       end
   end
```

params[:user][:login] == "xxxx 'OR' '1'='1'"

params[:user][:password] == "xxxx 'OR' '1'='1'"

Impact de l'attaque au niveau de l'application

```
class UsersController < ApplicationController
 def login
    if request.post?
       # whole user is stored in the session
       if session[:user] = User.authenticate(params[:user][:login],
params[:user][:password])
          flash[:notice]  = 'You have been successfully logged in.'
          if session[:user].admin
            redirect_to :controller => '/admin/home', :action => 'index'
          else
            redirect_to :controller => '/user/home',  :action => 'index'
          end
       else
          flash.now[:error] = 'Login failed'
       end
   end
```

params[:user][:login] == "xxxx 'OR' '1'='1'"

params[:user][:password] == "xxxx 'OR' '1'='1'"

session[:user][:login] == "admin"

Impact de l'attaque au niveau de l'application

```
class UsersController < ApplicationController
 def login
    if request.post?
       # whole user is stored in the session
       if session[:user] = User.authenticate(params[:user][:login],
params[:user][:password])
          flash[:notice]  = 'You have been successfully logged in.'
          if session[:user].admin
            redirect_to :controller => '/admin/home', :action => 'index'
          else
            redirect_to :controller => '/user/home',  :action => 'index'
          end
       else
          flash.now[:error] = 'Login failed'
       end
   end
```

params[:user][:login] == "xxxx 'OR' '1'='1'"

params[:user][:password] == "xxxx 'OR' '1'='1'"

session[:user][:login] == "admin"

params[:user][:login] != session[:user][:login]

Détection possible

```
class UsersController < ApplicationController
 def login
    if request.post?
      # whole user is stored in the session
      if session[:user] = User.authenticate(params[:user][:login],
params[:user][:password])
        flash[:notice]  = 'You have been successfully logged in.'
        if session[:user].admin
          redirect_to :controller => '/admin/home', :action => 'index'
        else
          redirect_to :controller => '/user/home',  :action => 'index'
        end
      else
        flash.now[:error] = 'Login failed'
      end
  end
```

params[:user][:login] == session[:user][:login]

Construction des invariants

- Utilisation d'un outil externe appelé Daikon.
- Développé au MIT par l'équipe Program Analysis Group http ://groups.csail.mit.edu/pag/.
- Outil issu du monde du test logiciel.
- Destiné à vérifier la conformité d'un logiciel vis-à-vis de sa spécification.

Daikon

- Permet le calcul d'invariants potentiels.
- Algorithme *Generate and check*
- À partir d'une trace contenant par exemple le contenu de variables à certains instants de l'exécution, des invariants potentiels sont dérivés.
- Ils sont invalidés au fur et à mesure du traitement des traces.
- Au final ne sont conservés que les invariants vrais pour toutes les traces

Réduction du nombre d'invariants

- Afin de diminuer le nombre d'invariants (potentiels), on ne s'intéresse qu'aux variables qui dépendent des entrées utilisateur.
- Ces variables sont les seules dont les valeurs peuvent être influencées par l'utilisateur.
- Afin de traquer ces variables, on se base sur la notion de *taint checking*.
- Propagation d'une marque apposée sur les entrées via le flot de données.
- Partiellement supportée par l'interpréteur Ruby.

Apprentissage automatique

États de l'application

- Plugin RoR indépendant de l'application considérée
- Préempte l'exécution de l'application à chaque instruction
- Inspecte toutes les variables de l'application
- Conserve l'état des variables dépendant des entrées utilisateurs

Apprentissage automatique

Code à exécuter

```
def action
```

```
instr1
```

```
instr2
```

```
instr3
```

```
end
```

Apprentissage automatique

Code à exécuter

```
def action
```

```
instr1
```

```
instr2
```

```
instr3
```

```
end
```

Exécution réelle

```
def func
```

```
getTaintedVars()
SaveTaintedVars()
```

```
instr1
```

```
getTaintedVars()
SaveTaintedVars()
```

```
instr2
```

```
getTaintedVars()
SaveTaintedVars()
```

```
instr3
```

```
getTaintedVars()
SaveTaintedVars()
CreateTraceFile()
```

```
end
```

Apprentissage automatique

Parcours automatique de l'application

- Nécessité de parcourir l'application
- Permet de couvrir l'ensemble des cas *normaux*
- Utilisation d'outils de test fonctionnels : Selenium

Traces générées

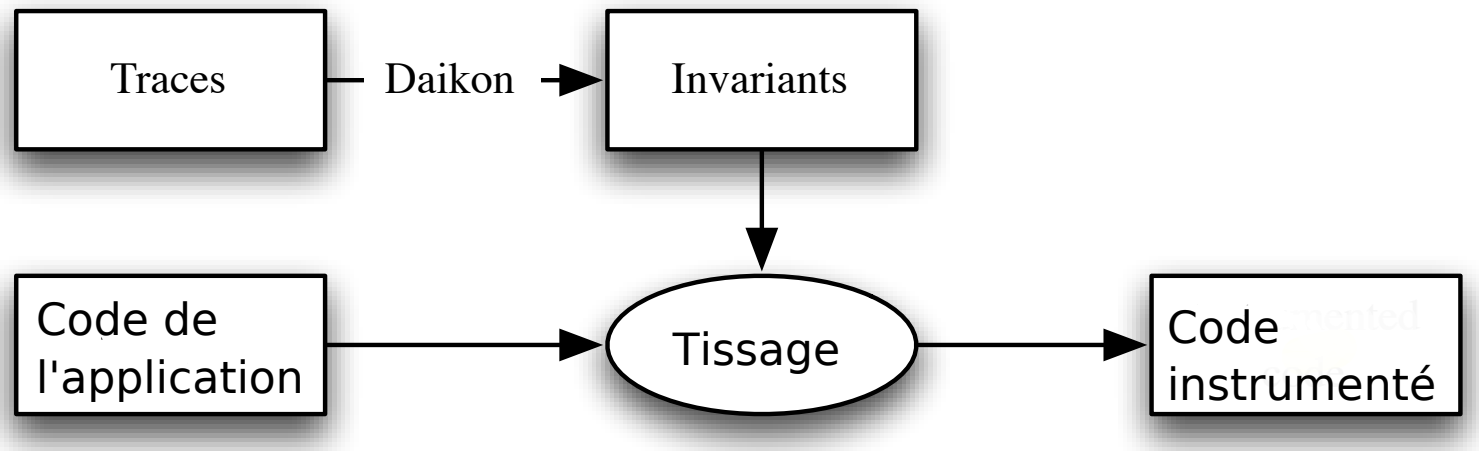
```
variable 17_1_@_params["user"]["login"]
  var-kind variable
  dec-type String
  rep-type java.lang.String
  enclosing-var 17_1_@_params["user"]
  comparability 0
variable 19_2_@_session["user"].@attributes["login"]
  var-kind variable
  dec-type String
  rep-type java.lang.String
  enclosing-var 19_2_@_session[user].@attributes
  comparability 0
...
17_1_@_params["user"]["login"]
"evette"
1
19_2_@_session["user"]@attributes["login"]
"evette"
1
```

Traces générées

```
variable 17_1_@_params["user"]["login"]
  var-kind variable
  dec-type String
  rep-type java.lang.String
  enclosing-var 17_1_@_params["user"]
  comparability 0
variable 19_2_@_session["user"].@attributes["login"]
  var-kind variable
  dec-type String
  rep-type java.lang.String
  enclosing-var 19_2_@_session[user].@attributes
  comparability 0
...
17_1_@_params["user"]["login"]
"evette"
1
19_2_@_session["user"]@attributes["login"]
"evette"
1
```

```
17_1_@_params[user][name] == 19_2_@_session[user].@attributes[name]
```

Tissage


Traces
Daikon
Invariants
Code de l'application
Tissage
Code instrumenté

Code du contrôleur sensible à une attaque par injection SQL

```
class UsersController < ApplicationController
 def login
    if request.post?
       # whole user is stored in the session
       if session[:user] = User.authenticate(params[:user][:login],
params[:user][:password])
          flash[:notice]  = 'You have been successfully logged in.'
          if session[:user].admin
            redirect_to :controller => '/admin/home', :action => 'index'
          else
            redirect_to :controller => '/user/home',  :action => 'index'
          end
       else
          flash.now[:error] = 'Login failed'
       end
   end
```

Code instrumenté automatiquement

17_1_@_params[user][name]

```
def login
 if request.post?
  if isDefined?('@_params["user"]["login"]') #AutoGenerated
   IDSAsserts.store('17_1_@_params["user"]["login"]',@_params["user"]["login"])
  end
  # whole user is stored in the session
  if session[:user] = User.authenticate(params[:user][:login],
   params[:user][:password])
  if isDefined?('@_session[:user]["login"]') #AutoGenerated
   IDSAsserts.store('19_2_@_session[:user"][:login]',@_session[:user][:login])
   assertEqualVar('17_1_@_params["user"]["login"]',
    '19_2_@_session[:user][:login]')
  end
  flash[:notice]  = 'You have been successfully logged in.'
  if session[:user].admin
   redirect_to :controller => '/admin/home', :action => 'index'
   else
    redirect_to :controller => '/user/home',  :action => 'index'
   end
  else
   flash.now[:error] = 'Login failed'
  end
 end
end
```

Code instrumenté automatiquement

19_2_@_session[user].@attributes[name]

```
def login
 if request.post?
  if isDefined?('@_params["user"]["login"]') #AutoGenerated
   IDSAsserts.store('17_1_@_params["user"]["login"]',@_params["user"]["login"])
  end
  # whole user is stored in the session
  if session[:user] = User.authenticate(params[:user][:login],
   params[:user][:password])
  if isDefined?('@_session[:user]["login"]') #AutoGenerated
   IDSAsserts.store('19_2_@_session[:user"][:login]',@_session[:user][:login])
   assertEqualVar('17_1_@_params["user"]["login"]',
    '19_2_@_session[:user][:login]')
  end
  flash[:notice]  = 'You_have_been_successfully_logged_in.'
  if session[:user].admin
   redirect_to :controller => '/admin/home', :action => 'index'
   else
    redirect_to :controller => '/user/home',  :action => 'index'
   end
  else
   flash.now[:error] = 'Login_failed'
  end
 end
end
```

Code instrumenté automatiquement

Vérification de l'assertion

```
def login
 if request.post?
  if isDefined?('@_params["user"]["login"]') #AutoGenerated
   IDSAsserts.store('17_1_@_params["user"]["login"]',@_params["user"]["login"])
  end
  # whole user is stored in the session
  if session[:user] = User.authenticate(params[:user][:login],
   params[:user][:password])
  if isDefined?('@_session[:user]["login"]') #AutoGenerated
   IDSAsserts.store('19_2_@_session[:user"][:login]',@_session[:user][:login])
   assertEqualVar('17_1_@_params["user"]["login"]',
    '19_2_@_session[:user][:login]')
  end
  flash[:notice]  = 'You have been successfully logged in.'
  if session[:user].admin
   redirect_to :controller => '/admin/home', :action => 'index'
   else
    redirect_to :controller => '/user/home',  :action => 'index'
   end
  else
   flash.now[:error] = 'Login failed'
  end
 end
end
```

Code instrumenté automatiquement

17_1_@_params[user][name] == 19_2_@_session[user].@attributes[name]

```
def login
 if request.post?
  if isDefined?('@_params["user"]["login"]') #AutoGenerated
   IDSAsserts.store('17_1_@_params["user"]["login"]',@_params["user"]["login"])
  end
  # whole user is stored in the session
  if session[:user] = User.authenticate(params[:user][:login],
   params[:user][:password])
  if isDefined?('@_session[:user]["login"]') #AutoGenerated
   IDSAsserts.store('19_2_@_session[:user"][:login]',@_session[:user][:login])
   assertEqualVar('17_1_@_params["user"]["login"]',
    '19_2_@_session[:user][:login]')
  end
  flash[:notice]  = 'You have been successfully logged in.'
  if session[:user].admin
   redirect_to :controller => '/admin/home', :action => 'index'
   else
    redirect_to :controller => '/user/home',  :action => 'index'
   end
  else
   flash.now[:error] = 'Login failed'
  end
 end
end
```

Résultats obtenus sur l'ensemble du code

Fichier	Nombre de lignes initial	Nombre de lignes du code instrumenté
application_controller.rb	20	20
home_controller.rb	11	537
products_controller.rb	26	1503
reviews_controller.rb	37	1547
users_controller.rb	51	1681

Temps moyen d'exécution :

- Avant instrumentation : 16.5*ms*.
- Après instrumentation : 125*ms*.

Conclusion

- Résultats encourageants à l'évaluation.
- Encore des améliorations à apporter au niveau de l'apprentissage (diminution du taux de faux positifs).
- Optimisation de l'implémentation éventuelle afin de diminuer le surcoût à l'exécution.

RRABIDS: un système de détection d'intrusions comportemental destiné aux applications Web écrites en Ruby on Rails.

Romaric Ludinard, Loïc Le Hennaf, Éric Totel

Supélec

Vendredi 10 juin 2011