Industrialiser les micro-changements :

la gestion des patchs

Yannick FOURASTIER
RESIST - ENSEEIHT - Mai 2004
yfourastier@antheya.com

Introduction
"Pourquoi gérer des patchs ?"

Sommaire

- Un **micro changement de version** pour quoi faire ? Du point de vue sécurité :
  - ***Renforcer*** : fonctionnalité nouvelle, procédé plus resistant
  - ***Corriger*** une vulnérabilité

- **Sûreté** : fiabilité *vs* sécurité, garantir l'état d'un *Système*
  - limiter les ***défaillances***
  - maîtriser les ***dysfonctionnements***

↳ **Capacité d'action**

- évaluation des besoins d'évolutions, priorités, "effets secondaires", planification, etc.

- **Sécurité** : gestion des risques
  - mesurer l'exposition (traitement des vulnérabilités) : injections de codes, dénis, détournements, ou extrusion de données
  - contrôler et réduire l'exposition, surtout en cas de menace avérée (vers, pandémie)

↳ **Capacité de réaction : disposer d'un process industrialisé**

- une démarche rationalisée et sous contrôle, pas nécessairement automatisée

Procédés
"Comment opérer ?"

Sommaire

- La gestion des *mises à jour* sous-tend **trois opérations** :
  - **Qualifier :** quel *patch* pour quelle *mise à niveau* ?
  - **Distribuer :** *où* et *comment* amener la mise à niveau nécessaire ?
  - **Appliquer :** *quand* et *comment réaliser* la mise à niveau ?

- **La qualification des patchs :**
  - **Objectif :** maîtriser l'ensemble de la *mise à niveau* !
    - référencer le patch et connaitre ses caractéristiques (but : maintenir la sûreté)
    - identifier les pré-requis et les conséquences du changement de version : attention aux impacts de modification logicielle !
    - définir les instructions pour l'application du patch (ou l'accompagnant).
  - Cette opération s'inscrit dans un processus :
    - "Veille sécurité" pour tout Utilisateur (individu ou "entité"),
    - "Recherche appliquée" pour tout fournisseur, éditeur ou constructeur : cadre juridique de la "fiabilité", maintenances logicielles, conditions de garanties, etc.

      ↳ vaste programme : responsabilités, LEN, aspects du "disclosure", etc.

Procédés

Distribution : méthodes *push* / *pull*

Sommaire

- Un **lot** de *mise à niveau* est constitué :
  - du fichier "en clair" (patchs de sources) ou binaire (exec, libs) à appliquer
  - des instructions de mise en oeuvre,
  - éventuellement des informations sur les impacts de la mise à niveau

- La **diffusion** des *mises à niveau* :
  - **Objectif :** rendre le *patch* disponible sur le lieu d'application
  - Deux méthodes pour un problème de logistique :
    - ***Pull :*** amener le *patch* sur le lieu d'application
    - ***Push :*** aller chercher le *patch* depuis le lieu d'application
  - Un problème de sécurité :
    - vulnérabilisation du s*ystème*
    - délais de mise à disposition du *patch*

- La **réalisation** de la *mise à niveau* :
  - **Objectif :** appliquer le *patch* en respectant les instructions de mise à niveau
  - Réalisation effective de la mise à niveau
  - *Roll-back* pour annuler le changement si doute ou problème (Murphy est joueur…)
  - Enregistrement du changement de version en cas de réussite

Industrialisation

Rappel : "En quoi ça consiste?"

Sommaire

- **Traitement** industrialisé :
  - Défini : procédé rationalisé et reproductible
  - Systématique : procédures standardisées et généralisées sur un périmètre délimité
  - Contrôlé : métrologie pour mesurer le fonctionnement / les dérives et optimiser

↳ **Fonctionnement systémique**

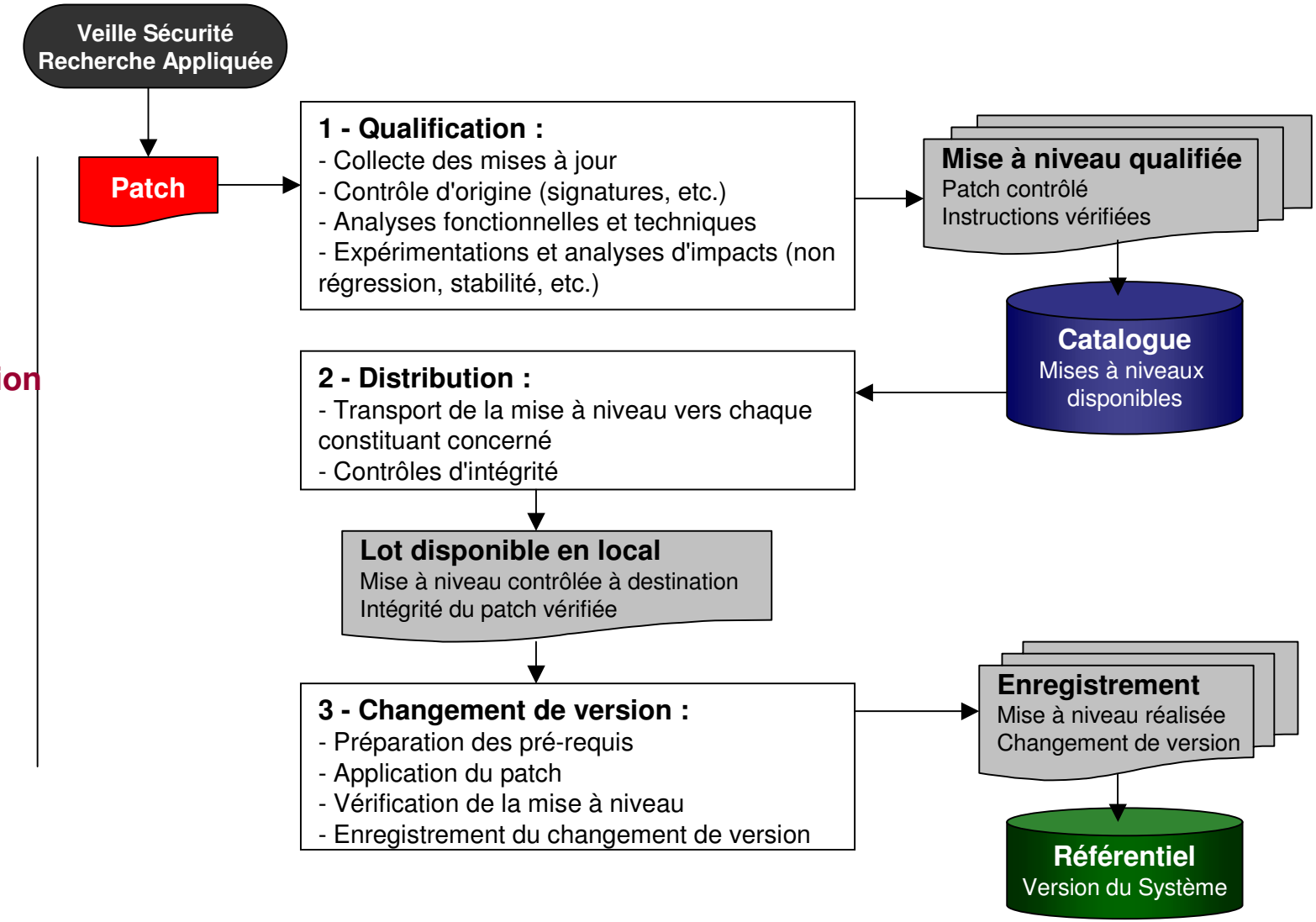
- Pour la "gestion des patchs" : déployer des **changements de version**
  - Distribuer uniquement des mises à niveau qualifiées
  - Appliquer systématiquement les patchs sur les cibles désignées
  - Contrôler le respect de chacune des étapes
  - Enregistrer le changement de version du système en fin d'opération

↳ **Maintien d'un système complexe à niveau de version référencé**

Industrialisation

Canevas d'opérations

Sommaire



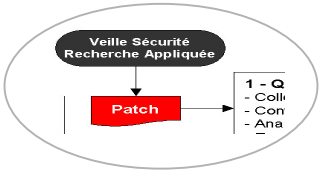
Précautions

Quels risques dans le traitement ?

Sommaire

- Risques sur **la mise à jour** :
  - **Risque 1** : elle est **vulnérable** → le problème de faille est seulement déplacé
    - ↳ **exemples multiples :** Windows, Unix, Applicatifs, ERP, etc. Les plus médiatiques :
      - SP1 de WinNT4 (1998), patch eEye pour IIS (2002/2003), etc.
      - Solaris 2.6 (kernel patch 1999), SGI (2003), Cisco IOS/CatOS
      - Webservices (C/C++/C#) : de MySAP à Weblogic, Websphere, Coldfusion, etc.
  - **Risque 2** : elle est **corrompue** → Trojan, bombe logique, etc.
    - ↳ **exemples marquants :** Serveurs Debian pour les m.à.j Woody (piratage 2003), Cas Oracle (updates Oracle Applications 2002), interrogations sur Microsoft … Backdoors Cisco IOS (cf. recherches de FX en 2003)
  - **Risque 3** : elle **impacte** la sûreté de fonctionnement → dénis, dysfonctionnements, régressions fonctionnelles, etc.
    - ↳ **exemples génants :** SP3 WinNT4 (1999), firmwares SAN Brocades (2002), releases IOS/CatOS
- Risques sur **le process** :
  - **Risque 4** : **environnement** corrompu → peut on avoir confiance sur la mise à niveau qui se retrouve disponible sur le lieu d'application ?
  - **Risque 5** : rigueur du **traitement** → réalité du micro changement de version déclenché

Précautions
La mise à jour est elle bonne ?

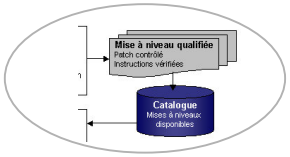


Sommaire

- **Précautions sur la mise à jour** "trouvée" ou codée :
  - **contrôle** de signature MD5 fournie.
    - Exemple sur une mise à jour (rsync, 1er mai 2004) :
    - Localisation : http://security.debian.org/pool/updates/main/r/rsync/rsync_2.5.5-0.4_i386.deb
    - Signature annoncée (Security advisory) : 5a5c172e1700f94050cd9b11482861d8
    - Commande :

```
yris@vignemale:~$ md5sum  rsync_2.5.5-0.4_i386.deb
5a5c172e1700f94050cd9b11482861d8  rsync_2.5.5-0.4_i386.deb
```

  - **analyse**, suivant la sensibilité de la cible à *updater* :
    - Sources : audit de code
    - Binaires : *reverse* et étude de code assemblé
  - **étude** d'impact du micro-changement de version : fiabilité, pré-requis ?
- **Précautions pour le process,** à la qualification :
  - **environnement** de qualification : sain, protégé, contrôlé
  - **facteur humain** : rigueur et fiabilité
  - **stockage** de la mise à niveau qualifiée
  - **anomalie détaillée** si non qualification !
  - **information** de disponibilité de la mise à niveau (log, main courante, etc.)

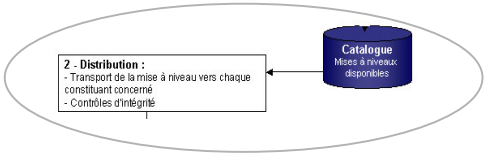


Précautions

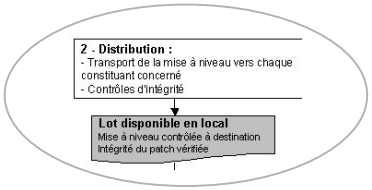
La bonne mise à jour arrive à destination ?

Sommaire

- **Précautions sur la mise à jour** :
  - **Scellement** du lot de mise à niveau
  - **contrôles** d'intégrité { fichier ; instructions }



- **Précautions pour le process,** à la distribution :
  - **stockage** au point de distribution : lecture seule dés la mise à disposition
  - **stockage** au point d'utilisation : droits *owner* ≠ droits *user* (*read only*)
  - **transport :** cheminement connu (segments et noeuds identifiés)
  - **flux :** scellé + contrôle d'intégrité



↳ **Rigueur du scellement {fichiers ; flux}**

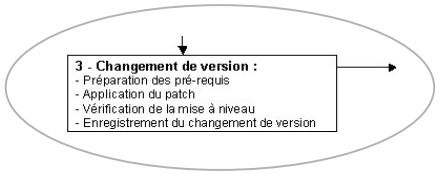
↳ **Importance des opérations de contrôle d'intégrité**

↳ **Zèle inutile : la continuité de sceaux suffit**
→ rationalisation des opérations (propriétés exploitées : transitivité et surjectivité)

Précautions

L'application de la mise à jour : effective ?

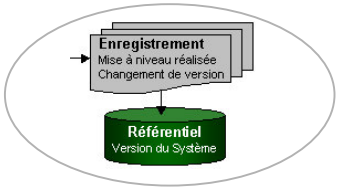
Sommaire



- **La mise à niveau est un lot composé de :**
  - instructions de **préparation :**
    - le socle à modifier est bien celui prévu
    - les pré-requis sont réunis : versions d'éléments connexes, configurations, etc.
  - instructions d'**application**
  - instructions de **vérification**
  - **fichier** de version différente du fichier à remplacer, ou impactant sa version (patch)

- **Précautions sur le traitement de mise à niveau :**
  - intégrité du traitement : toutes les instructions définies **doivent** être réalisées
  - **possibilité d'annuler** les modifications en cas d'anomalie :
    - sauvegarde des éléments à modifier
    - procédure de *roll back* prévue
  - **vérification** par tests des résultats des opérations réalisées

    ↳ **exemple :** md5 sur le fichier patché, calcul tripwire avant/après
    MBSA / HFNetCheck contrôle des HotFixes / Windows, etc.

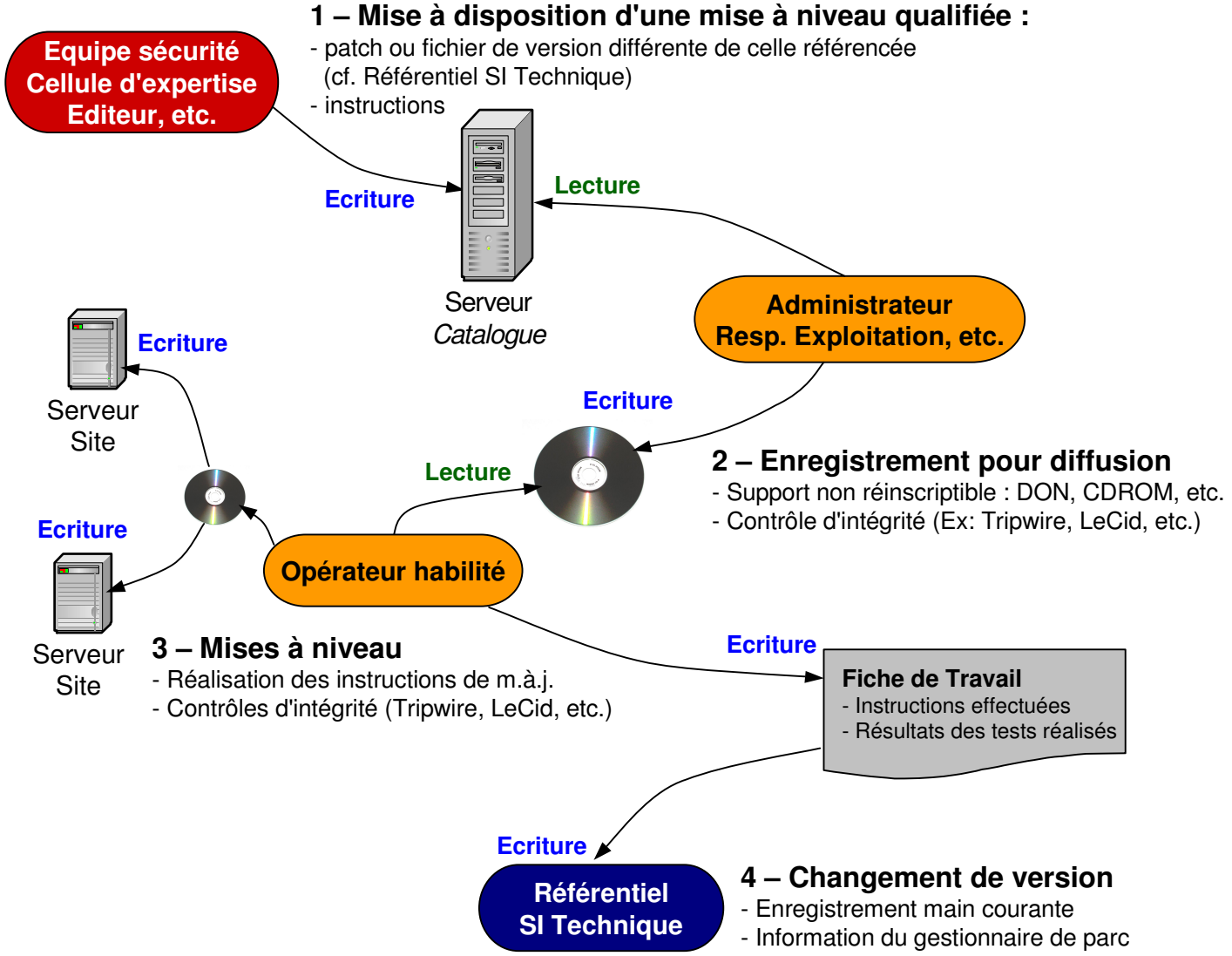


- **Enregistrer le changement de version**

Exemples

Système de mise à jour descendant manuel

Sommaire



Exemples

Packages : solaris, apt, rpm ou windowsupdate

Sommaire

- **Un package est :**
  - une **archive** contenant des données et/ou des programmes ainsi que les informations nécessaires à une **installation correcte** de ceux-ci sur le système.
  - constitué d'un et un seul fichier, ce n'est pas un exécutable, il est pris en charge par un programme dédié, le **gestionnaire de package**

- **Exemple Redhat Package Manager (rpm) :**
  - gestion de **version**, des **dépendances** et **scripts** d'installation / désinstallation
  - scripts de pré-installation, post-installation et pré-désinstallation et post-desinstallation
  - RPM/SOURCES, RPM/SPECS et fichier .spec Specfile normalisé
    - ↳ compilation rpm : RPM/RPMS

```
yris@canigou:~$ rpm -bb rsync-2.5.5-0.4_i386.spec
```

    - ↳ packages de sources : RPM/SRPMS

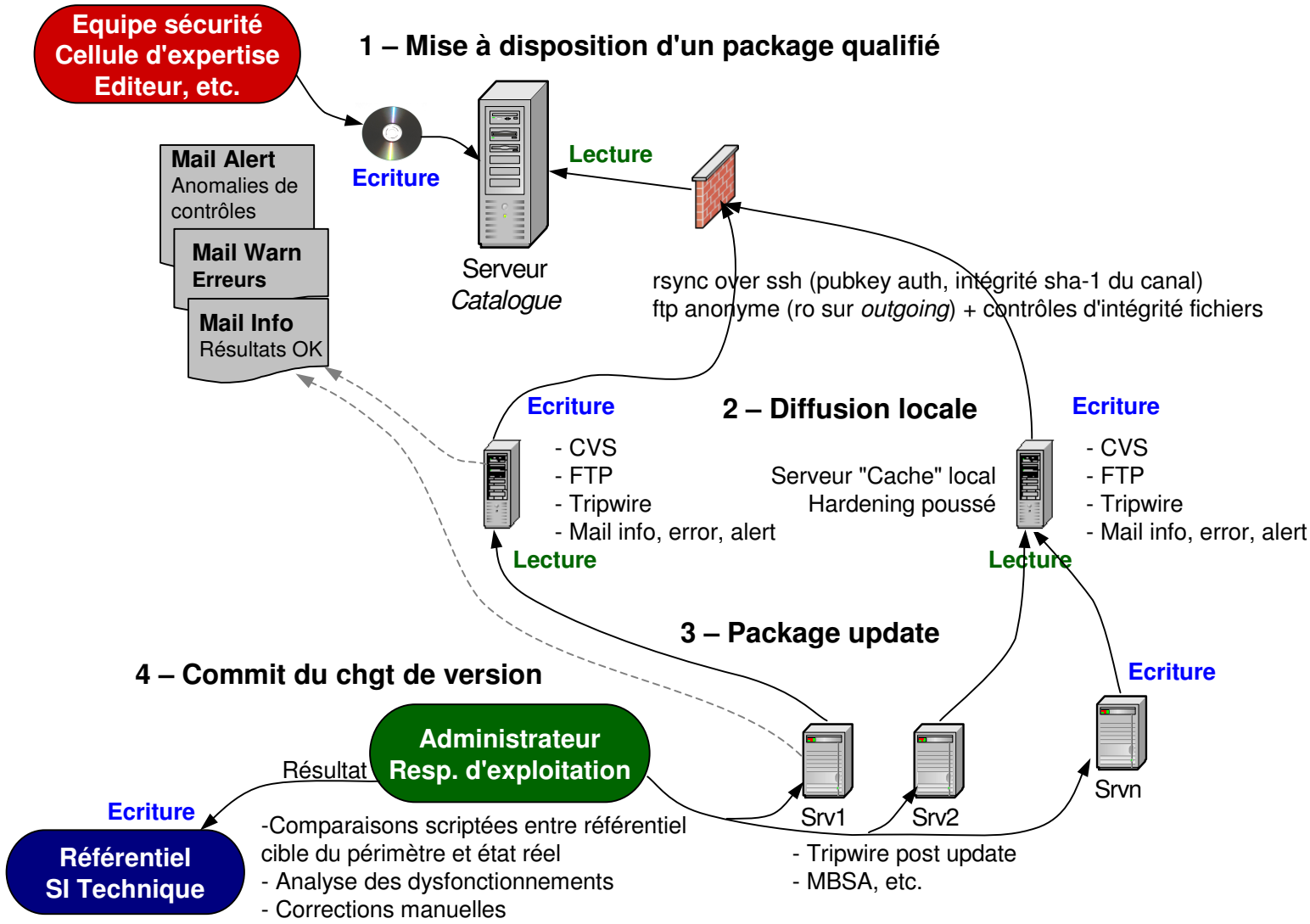
```
yris@canigou:~$ rpm -bs rsync-sources-2.5.5-0.4_i386.spec
```

- **Gestion par CD ou *online* (prudence : *cache poisonning* de proxies) :**
  - apt / dpkg / dselect : gestionnaire de package d'origine Debian
  - pkg(add / rm / chk / info) : gestionnaire de package Solaris
  - wupdmgr / Software Update Service + AD : gestionnaires de mise à jour Windows
    - ↳ **Spécification du serveur de mise à jour**
      hiérarchisation des sites de stockage, localisation des débits de màj

Exemples

Système *push* / *pull* basé CVS, FTPS

Sommaire



Conclusion

"savons nous administrer en version ?"

Sommaire

- Maîtriser les versions, ça doit permettre en instantané de :
  - **connaître** le niveau de sûreté
  - **contrôler** l'exposition aux risques

- Traiter la gestion des patchs : un "simple" problème de Qualité (*versionning*) !
  - maîtriser les mises à niveau (dont patchs !) à appliquer : les pré-requis, les correctifs ou modifications fonctionnelles et leurs impacts.
  - préparer les mises à jours qualifiées : lots d'instructions et fichiers patchs
  - distribuer / diffuser des instructions de changement et les fichiers associés,
  - appliquer des instructions
  - contrôler la conformité : fonctionnelle, sécuritaire… et la bonne application !

- Industrialiser : travailler dans la rigueur et la méthode d'un mode "process"
  - **Fiabiliser** les processus et les opérations rationalisées
  - **Systématiser** l'application des patchs
  - **Garantir** le niveau de version d'un Système donné

↳ **Viser un fonctionnement organisationnel et technique SYSTEMIQUE !**