Wapt – solution de déploiement logiciel sous licence GPLv3 (ou comment dire au revoir à SCCM)

Nathalie Vieira

WAPT – Solution de déploiement logiciel sous licence GPLv3 (ou comment dire au revoir à SCCM)

Nathalie Vieira
INRA CBGP
755, avenue du campus Agropolis
CS30016
34988 Montferrier-sur-Lez cedex

Jean-Charles Granger
Montpellier SupAgro
2, place Pierre Viala
34060 Montpellier Cedex 2

Résumé

Le centre INRA/Montpellier SupAgro regroupe une trentaine d'unités, 2500 agents de différents organismes de recherche et 1500 élèves de l'école d'ingénieurs, pour un total d'environ 3000 postes informatiques gérés par une quarantaine de gestionnaires de parc. Ce parc est diversifié du point de vue du matériel, des systèmes d'exploitation, et des logiciels dont la gestion est hétérogène.

Pour les clients Windows, qui constituent la majorité du parc, nous cherchions à proposer une stratégie de gestion des applications cohérente et centralisée, simple et peu coûteuse pour les gestionnaires. Cette solution devait être en adéquation avec notre environnement Active Directory sous licence GPL (Samba4). Nous avons choisi la solution WAPT (Windows APT-get) sous licence GPL v3.

La solution consiste à déposer les applications validées et packagées, garantissant ainsi leur conformité, sur un serveur dépôt (type dépôt apt) et permettant aux utilisateurs de les installer, via un portail web.

Elle intègre une console graphique de gestion centralisée (waptconsole) nous permettant de gérer les applications ou groupes d'applications (installation, mises à jour, configuration et désinstallation des applications, visualisation du statut des déploiements) et de les affecter aux postes enregistrés via un agent à installer sur les clients (waptagent).

Nous présentons ici la facilité de mise en place du serveur de dépôt, la simplicité de création des paquets et de leur configuration, les méthodes de déploiement de l'agent local et des applications, puis leur mise à jour sur les clients.

Mise en place depuis octobre 2016, la solution gère actuellement plus de 500 postes et propose plus d'une soixantaine d'applications validées.

Mots-clefs

WAPT, Déploiement logiciel, Windows, Python, configuration logiciels, mise à jour logiciels

1 Fonctionnement

1.1 Pourquoi WAPT

Qui n'a pas souhaité gérer les applications pour l'ensemble du parc Windows du domaine et hors domaine, de façon centralisée, par des serveurs Linux et des solutions sous licence libre (GPLv3) ? Voilà ce que permet WAPT.

Le centre INRA de Montpellier est fortement mutualisé avec l'école agronomique de Supagro tant en terme d'architecture serveur que ressource et compétence technique. Nous recherchions une solution homogène, simple et centralisée pour la gestion des logiciels à proposer à l'ensemble des gestionnaires de parcs si possible en adéquation avec note solution d'Active Directory sous licence GPL (Samba4).

La solution SSCM proposée par l'INRA a tout d'abord été étudiée puis écartée pour les raisons suivantes :

- Mise en place de serveurs dédiés INRA (AD et SCCM) ➡ besoin de maintenance et compétence supplémentaire pour l'administration des OS Windows Server
- Changement de nom domaine pour les postes INRA ➡ Multiplication de domaine et risque de confusion des utilisateurs
- Non prise en charge des postes hors domaines
- Mise en place longue et fastidieuse
- Premier contact avec WAPT

La solution WAPT répondait à un certain nombre de points de notre cahier des charges

- Mise en place de la solution sur des serveurs Linux
- Conservation du domaine MTP
- Prise en charge des postes hors domaine
- Sous licence GPLv3
- Mise en place et prise de mains rapides

1.2 Vue d'ensemble

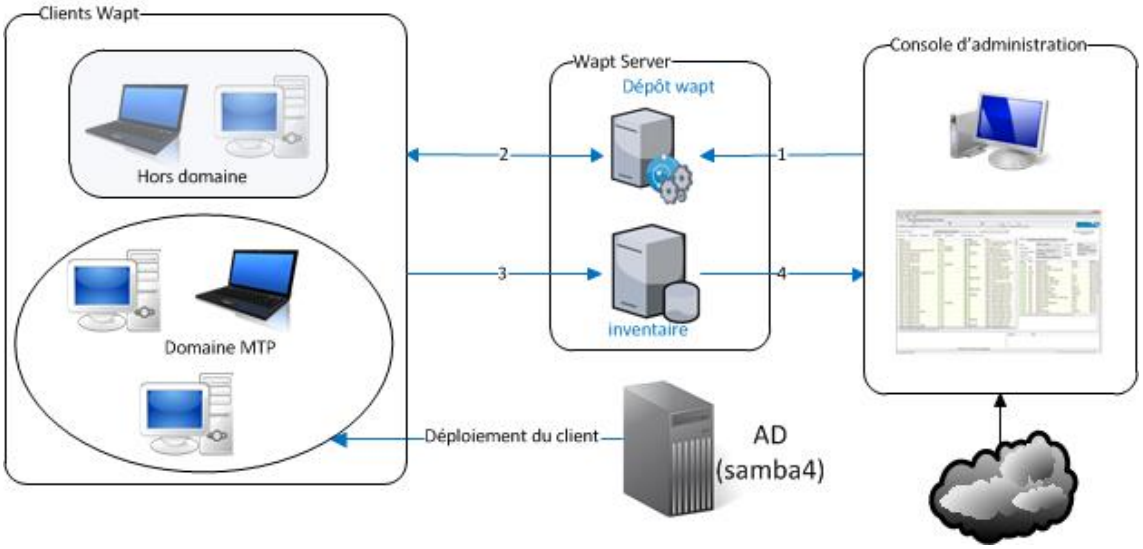

Figure 1 – Schéma d'ensemble

Inspiré par le fonctionnement du gestionnaire de paquet du système Debian apt-get, WAPT est développé (en Python) et maintenu par l'équipe de Tranquil IT System (1) depuis 2012.

Tout commence par la création d'un paquet WAPT ou l'importation depuis un dépôt externe *via* la console d'administration. Le paquet est déposé sur le serveur de dépôt local et est mis à disposition. Un fichier index nommé `Package`, accessible sur le serveur dépôt est également mis à jour.

Un `update`, exécuté par les clients (par défaut, toutes les 2 heures), télécharge le fichier index `Package` et le paquet `host` (défini ci-après), vérifie les mises à jour disponibles, les télécharge et les met en cache. Il envoie son statut d'inventaire au serveur.

L'arrêt de la machine, une action de l'administrateur ou de l'utilisateur ou une tâche planifiée déclenchent un `upgrade`. Un `upgrade` lance l'installation des paquets et les mises à jour, puis renvoie le statut d'inventaire au serveur.

1.3 Principe des paquets WAPT

De la même manière qu'un paquet de type Debian, le paquet WAPT embarque avec lui les binaires qui seront exécutés ainsi que les dépendances, et peut également contenir des fichiers de configuration. Avec le même principe de dépendance du gestionnaire de paquet APT, si le paquet WAPT nécessite l'installation d'un autre paquet WAPT au préalable, ce dernier sera installé en premier lieu.

Trois types de paquets existent :

1. *base* : paquet classique, contenant l'installeur et autres fichiers nécessaire à son fonctionnement

2. *host* : ce sont les paquets spécifiques à chaque machine, portant son nom FQDN (Fully Qualified Domain Name)
3. *group* : permettant de grouper les paquets de type *base* ou de type *host*

Le principe de dépendance commence par le paquet *host* que le client WAPT va chercher à installer. Pour installer ce paquet host, il devra satisfaire les dépendances indiquées dans celui-ci. Pour chaque dépendance, il devra satisfaire aux sous-dépendances et ainsi de suite.

Pour garantir la provenance d'un paquet, ce dernier est signé par l'administrateur du paquet et contient une somme de contrôle de la liste des fichiers contenus.

1.4 Architecture serveur

Le serveur WAPT repose sur trois rôles distincts et en partie dissociables :

1. Rôle de dépôt

Il permet de mettre à disposition les paquets WAPT sur un dépôt de fichiers web, et les installeurs *waptagent* (agent pour les postes client), *waptsetup* (pour l'installation de la console d'administration) et *waptdeploy* (pour le déploiement de l'agent via GPO). Ce rôle peut être assuré par un serveur Apache2 par exemple.

2. Rôle de serveur d'inventaire

Serveur passif collectant les informations que les agents WAPT lui transmettent. Il ne comporte pas d'interface web, seule la console d'administration peut permettre de consulter l'inventaire. Les agents WAPT transmettent l'inventaire matériel, logiciel et le statut des paquets WAPT. Ce rôle est assuré par un serveur MongoDB.

3. Rôle de Proxy Commande

Par son intermédiaire, les commandes poussées via la console d'administration sont transmises aux agents WAPT déployés.

2 Mise en place et présentation

La facilité et la rapidité de la mise en place sont un des nombreux avantages de la solution.

2.1 Serveur

WAPT n'a pas de prérequis contraignants pour pouvoir fonctionner. Nous avons choisi de le faire fonctionner sur une machine virtuelle XenServer légère : 1 CPU, 2 Go de RAM, 1 disque dur de 10 Go pour le système et 1 disque de 100 Go pour le stockage des paquets WAPT.

Le serveur peut être installé sur plusieurs systèmes d'exploitation (dont Windows) mais nous avons privilégié la simplicité en choisissant Debian 8 (Jessie). L'installation de WAPT est très simple, elle se base sur des paquets Debian parfaitement intégrés au système et n'ayant pas de dépendance exotique.

WAPT demande peu de ressources, il peut être installé sur une machine fournissant d'autres services. Cependant, pour des raisons de sécurité et de simplicité de configuration du serveur Apache, nous avons préféré héberger uniquement ce service.

2.2 Console d'administration

Nous sommes deux administrateurs WAPT, nous utilisons chacun une console d'administration que avons configurées à l'identique. Pour installer la console, un exécutable est disponible sur l'interface du serveur dépôt WAPT local, il se nomme `waptsetup.exe`. Après l'avoir lancé et indiqué l'URL de notre serveur WAPT et du serveur dépôt, nous lançons la console via l'exécutable `C:\wapt\waptconsole.exe`.

Afin de garantir la provenance des paquets, il est nécessaire de générer un couple clé privée/certificat qui signera la provenance des paquets du dépôt. Nous avons opté pour la génération d'un couple clé privée/certificat distinct par administrateur en utilisant le menu intégré dans la console d'administration. Les certificats publics sont ensuite diffusés sur tous les postes lors de l'installation de l'agent WAPT.

Nous avons pris soin de placer la clef privée `PrivateKey.pem` dans un dossier local protégé sur chaque poste utilisant la console. Les certificats publics sont placés dans le dossier `c:\wapt\ssl`. Il est important de ne pas laisser la clef privée dans le dossier `c:\wapt`, car il est utilisé lors de la création de l'agent déployé sur les postes des utilisateurs.

La console d'administration nous sert également lors de l'édition, la création et l'importation des paquets WAPT, nous devons alors finaliser sa configuration. Pour cela, deux méthodes existent, soit via la console d'administration soit en éditant le fichier de configuration `waptconsole.ini` dont nous donnons un exemple ci-dessous. Les points importants de cette configuration sont :

- le préfixe donné aux paquets (visibilité et uniformité pour les usagers) ;
- le chemin de la clef privée (permettant de signer les paquets) ;
- l'URL du serveur dépôt que les clients utiliseront comme base de référence.

```
[global]
waptupdate_task_period=120
WAPT_server=https://wapt.mydomain.fr
repo_url=http://wapt.mydomain.fr/wapt
use_hostpackages=1
last_usage_report=15/09/2017 10:07:33
http_proxy=
default_package_prefix=appstore
default_sources_root=C:\waptdev
private_key=C:\private\PrivateKey.pem
templates_repo_url=https://store.wapt.fr/wapt
use_http_proxy_for_templates=0
use_http_proxy_for_server=0
use_http_proxy_for_repo=0
send_usage_report=1
language=fr
advanced_mode=1
authorized_certs_dir=%appdata%\waptconsole\ssl
```

La dernière étape est la création de l'agent WAPT qui doit être généré par une des deux consoles et utilise le dossier `c:\WAPT`. Nous utilisons l'outil disponible depuis la console, en s'assurant que les deux certificats publics sont présents sous `c:\wapt\ssl`. La fin du processus de génération de l'agent charge ce dernier sur le serveur WAPT, le rendant disponible *via* l'exécutable `waptagent.exe`.

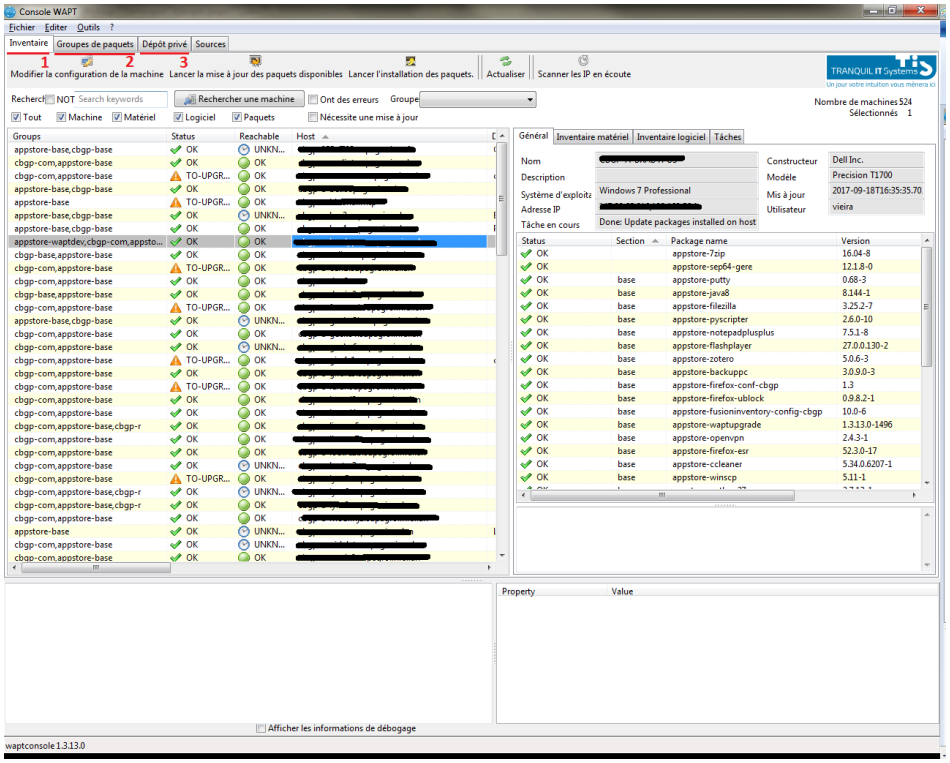

Figure 2 - Console d'administration

La console d'administration nous permet de visualiser l'ensemble des clients disposant de l'agent WAPT s'étant enregistrés sur le serveur. Elle est composée de trois parties:

1. Inventaire : La partie de gauche liste l'ensemble des clients, avec leur statut (client joignable ou non et ensemble des applications installé sur le client à jour ou non) et le(s) groupe(s) de paquet(s) que nous leur avons assigné. La partie de droite comporte un détail de l'inventaire logiciel ainsi que leur statut si ces logiciels ont été installés via WAPT, et l'inventaire matériel remonté lors de l'installation.
2. Groupe de paquets : liste l'ensemble des groupes et des logiciels associés. Un paquet *host* peut être associé à plusieurs paquets *group*.
3. Dépôt privé : liste l'ensemble des paquets disponibles sur le serveur dépôt WAPT. Cette partie nous permet d'importer les paquets depuis internet ou un fichier et de les éditer.

2.3 Client

Nous utilisons l'exécutable `waptdeploy.exe` et son checksum fournit par le serveur de dépôt pour le déploiement via GPO. Ce dernier va tester la présence de l'agent WAPT et sa version, télécharger au besoin l'agent `waptagent.exe` depuis le serveur dépôt, et lancer l'installation silencieuse avec les paramètres prédéfinis lors de la génération de l'agent.

Le même agent `waptagent.exe` est installé manuellement sur les postes hors domaine (postes personnels ou professionnels hors domaine). Le fonctionnement général reste identique.

Les clients disposent ainsi d'un panel d'exécutables et de services distincts :

1. le service `waptservice` : il écoute sur le port 8088, exécute les actions transmises par la console d'administration, et fournit une interface web locale aux utilisateurs (utilisée pour le `self-service` et destinée aux utilisateurs non administrateur de leur poste afin qu'ils puissent installer les applications proposées dans le catalogue WAPT), qui permet de visualiser les applications, de les installer, les supprimer, et les mettre à jour manuellement ;
2. l'exécutable `wapt-get.exe` : il a pour rôle de lancer les actions (update, upgrade, install, remove, etc.) depuis la ligne de commande ;
3. la librairie Python setuphelpers : elle inclut toutes les fonctionnalités de WAPT pour l'installation d'une application ;
4. l'exécutable `wapttray.exe` : il ajoute dans la zone de notification une icône d'information sur l'état du client et propose les mises à jour manuellement. Nous avons choisi de ne pas utiliser cette fonctionnalité.
5. l'exécutable `waptexit.exe` : il permet, lors de l'arrêt du poste, de proposer aux utilisateurs d'effectuer les mises à jour.

L'installation, les mises à jour et la désinstallation peuvent ainsi s'effectuer de plusieurs façons :

- poussées par le serveur via la console d'administration par le service `waptservice`,
- proposées au client à l'arrêt ou au redémarrage du poste par la commande `waptexit.exe`,
- manuellement, via un exécutable `waptray.exe`, un portail web (http://127.0.0.1:8088 – service waptservice)
- par ligne de commande `wapt-get.exe`.

Par défaut et à l'initiative de `waptservice`, un update est effectué toutes les deux heures.

3 Gestion des paquets

La création des paquets est facilitée, sans être expert du langage Python, via l'ajout d'une librairie Python, `setuphelpers` et des outils adaptés.

3.1 Composition

Voici un exemple de composition d'un paquet WAPT *a minima* :

```
appstore-backuppc
├── cygwin-rsyncd-3.0.9.0_installer.exe
├── setup.py
└── WAPT
    ├── control
    ├── manifest.sha1
    ├── signature
    └── wapt.psproj
```

Nous retrouvons à la racine du paquet notre programme à installer `cygwin-rsyncd-3.0.9.0_installer.exe` et un fichier `setup.py`, qui contient les instructions interprétées par l'agent WAPT pour procéder à l'installation (voire la configuration) de ce dernier.

Dans le dossier WAPT

- le fichier `control` est la carte d'identité du paquet, il contient son nom, sa version, sa description, ses dépendances, l'empreinte de la signature du paquet, le nom du mainteneur ainsi que d'autres informations ;
- le fichier `manifest.sha1` contient la somme de contrôle pour l'ensemble des fichiers présents dans le paquet ;
- le fichier `signature` valide la provenance du paquet grâce aux certificats publics diffusés via l'agent ;

- le fichier `wapt.psproj` est utilisé lors de l'édition du paquet avec l'utilisation du RAD (Rapid Application Development) Pyscripter (préconisé par Tranquil IT).

Tous les paquets WAPT sont ainsi facilement transportables et lisibles. À noter qu'un paquet WAPT est un fichier zip.

3.2 Création

Les premiers paquets mis en place ont été importés depuis des dépôts externes, notamment le serveur de dépôt public de Tranquil IT, via la console d'administration. Le processus se déroule en quatre étapes :

1. Importation du paquet WAPT depuis un dépôt externe et modification du préfixe
2. Vérification du script d'installation (setup.py) pour assurer une conformité, si besoin modification du script.
3. Signature du paquet avec notre propre clef privée.
4. Envoi sur notre dépôt.

La solution WAPT propose un paquet (waptdev) pour la mise en place de l'environnement de développement de paquet. Ce dernier installe sur la machine de développement le RAD Pyscripter et Python 2.7.

Elle fournit également les outils pour la création de modèle de paquet soit par la console d'administration soit par ligne de commande (WAPT-get make template) à partir d'un installeur .msi ou .exe. Ceci aura pour conséquence :

1. la création d'un dossier nommé *prefix-nom_paquet*.
2. la copie de l'installeur à la racine de ce dossier.
3. la création d'un fichier setup.py (à la racine), d'un fichier control et wapt.psproj dans le sous dossier WAPT.

```
# -*- coding: utf-8 -*-
from setuphelpers  import *

uninstallkey = []

def install():
    print('installing appstore-mendeley')
    install_exe_if_needed("Mendeley.exe",'/S',key='
Mendeley Desktop',min_version='
1.17.10',accept_returncodes=[1223])
```

WAPT est édité avec le langage de programmation Python, et est livré avec une librairie `setuphelpers` simplifiant les actions fréquemment utilisées dans des fonctions prédéfinies (exemple : `install_exe_if_needed()`). À celles-ci nous pouvons ajouter (import) les fonctions usuelles du langage Python.

Trois grandes fonctions existent dans l'interprétation du fichier `setup.py` par l'agent WAPT :

1. `def_install()` : utiliser lors de l'installation du paquet
2. `def uninstall()` : utiliser lors de la désinstallation personnalisée du paquet
3. `def session_setup()` : permet de personnaliser un contexte utilisateur

La variable `uninstallkey` est importante et permet, lors de l'installation du paquet, de vérifier qu'elle coïncide avec la clef de registre utilisée par l'installeur. Certaines fonctions contenues dans la librairie `setuphelpers` (exemple `install_exe_if-needed`) l'attendent en argument.

Les paquets WAPT peuvent ne pas contenir d'installeur, et sont pour nous, des paquets de configuration dans lesquels chaque unité du centre peut personnaliser son environnement. Ainsi, nous diffusons un paquet principal contenant l'installeur, puis nous ajoutons un paquet supplémentaire contenant une configuration personnalisée selon la politique informatique de l'unité. Le paquet de configuration a pour dépendance le paquet contenant l'installeur.

Voici un exemple de script `setup.py` – paquet `appstore-firefox-conf-cbgp`

```
from setuphelpers import *
import time

uninstallkey = []

def install():
    print(u"Copie des fichiers de configuration -
CBGP")
    copytree2(r'conf_firefox',r'c:\\Program Files
(x86)\\Mozilla Firefox\\',onreplace=default_overwrite)
    time.sleep(3)

def session_setup():
    print(u"Raccourcis Install Software")
    create_user_desktop_shortcut('Install software for
%s'%get_current_user()',
        target =
'http://127.0.0.1:8088/list',icon='wapt.ico')
```

Le script va copier la configuration du navigateur Firefox (indiqué comme dépendance dans le fichier control) pour cette unité et ajouter un raccourci sur le bureau

La création du paquet achevée, il ne reste plus qu'à le "construire" (signature avec notre clef privée, ajout du fichier `manifest.sha1` et compression au format ZIP) et à l'uploader sur le serveur dépôt, le rendant automatiquement disponible à tous les clients. Pour cela nous pouvons utiliser `Pyscripter` ou la ligne de commande `wapt-get`

`build-upload`.

3.3 Mise à jour

Le processus de mise à jour d'un paquet WAPT se déroule en 5 étapes :

1. importer le paquet sur l'environnement de développement
2. remplacer l'installeur vérifié
3. modifier si nécessaire le fichier `setup.py`
4. modifier le champ version du fichier `control`
5. signer le paquet et l'exporter

L'agent WAPT effectue un update toutes les deux heures, puis propose la mise à jour à l'utilisateur qui peut l'appliquer manuellement via l'interface web, l'exécutable `wapttray.exe` ou la ligne de commande ou encore lors de l'arrêt du poste.

En cas de mise à jour critique (par exemple pour VLC ou CCleaner pour corriger une faille de sécurité), nous pouvons pousser l'application de la mise à jour via la console d'administration.

3.4 Suppression

Lors de la désinstallation d'un paquet, l'agent local va rechercher la variable `UninstallKey` dans la clef de registre `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall`, puis exécuter la chaîne `UninstallString` afin de lancer le processus de désinstallation. Le principe est identique au fonctionnement utilisé par Windows.

Si le paquet contient des actions sous la fonction `def uninstall()`, elles seront exécutées.

Les canaux des demandes pour la suppression des paquets restent identiques, à distance (console d'administration) ou localement (interface web, exécutable `wapttray.exe` ou ligne de commande).

4 Retour d'expérience

La solution a été mise en place en octobre 2016, est gère actuellement plus de 500 postes dans différents laboratoires et salles d'enseignements. Nous proposons un catalogue de plus de 94 paquets (installeurs et configurations comprises). La solution nous permet :

- de conserver l'architecture mutualisée INRA Montpellier/Supagro
- d'administrer intégralement le cycle de vie applicatif de notre parc informatique, simplement et de façon centralisée ;
- de limiter les interventions aux seuls postes remontés en erreur dans la console ;
- de créer et configurer les paquets WAPT rapidement et aisément

- de forcer les mises à jour des applications posant un problème de sécurité (exemple : CCleaner et VLC)
- offre également, grâce au service `self-service`, une souplesse et une sécurité pour les utilisateurs finaux qu'ils soient administrateurs ou non.
- prochainement de désinstaller des applications non installées par la solution (exemple : logiciel constructeur DELL ou HP)

Un point fait actuellement défaut, vis-à-vis de notre structure informatique et administrative, il est lié à la délégation des droits au niveau des gestionnaires de parc : tous les administrateurs ayant accès aux deux consoles peuvent gérer l'intégralité des postes enregistrés même si ces derniers ne font pas partie de leur parc.Cette amélioration sera possiblement proposée dans la future version WAPT Corporate.

WAPT répond à notre besoin et est en adéquation avec notre politique informatique de choix des licences libres allié à une stratégie de sécurité pour les applications déployées.

Bibliographie

1. **Tranquil IT System.** s.l. : https://tranquil.it/.