Serveur winget privé (rewinged)

Source de paquets winget auto-hébergée pour le parc géré Make IT Simple. Permet d'aligner les versions logicielles sur l'ensemble des parcs clients sans donner à l'agent de déploiement le droit de remplacer son propre binaire.

1. Architecture

```
                                Poste client
                                     |
                                     | HTTPS 443
                                     v
                      +-----------------------------+
                      | VM Caddy (frontal existant)  |
                      | repo.makeitsimple.be         |
                      | TLS Let's Encrypt            |
                      +---------------+-------------+
                                      |
                                      | HTTP 8080 (interne)
                                      v
Forgejo                 +--------------------------------+
mis/winget-manifests    | VM Debian  winget-repo         |
  manifests YAML        |                                |
  + MSI en Git LFS      |  rewinged :8080                |
        |               |    -manifestPath  manifests    |
        | git pull      |    -autoInternalizePath        |
        | + lfs pull    |         installers             |
        | timer 1 min   |                                |
        +-------------->|                                |
                        +--------------------------------+
```

Composant	Machine	Rôle
Forgejo	existante	Source de vérité : manifests YAML et installeurs MSI.
Caddy frontal	frontal existant	Terminaison TLS. Aucun fichier local.
rewinged	VM dédiée	API REST `Microsoft.Rest` **et** service des installeurs.
Wazuh	existant	FIM sur manifests, installeurs et binaires.

Le dépôt Forgejo peut rester privé. Les postes ne dialoguent qu'avec `repo.makeitsimple.be`. Aucun accès anonyme à Forgejo n'est requis, puisque la synchronisation se fait par clé de déploiement SSH.

Chaîne de confiance. L'intégrité des paquets ne repose pas sur le TLS mais sur le SHA256 déclaré dans chaque manifest. Un installeur substitué échoue à la vérification et winget refuse l'installation.

2. Prérequis

- VM Debian 12 minimale, 1 vCPU / 1 Go RAM / disque dimensionné au volume d'installeurs
- Enregistrement DNS public `repo.makeitsimple.be` vers le frontal Caddy
- Dépôt Forgejo `mis/winget-manifests` avec Git LFS activé
- Clé de déploiement Forgejo **en lecture seule**
- Paquets `git` et `git-lfs` sur la VM

3. Installation de rewinged

3.1 Utilisateur et arborescence

```
useradd --system --home /srv/winget --shell /usr/sbin/nologin rewinged
mkdir -p /srv/winget
chown -R rewinged:rewinged /srv/winget
```

Les répertoires `manifests` et `installers` seront créés par le clone Git (section 6).

Le compte `rewinged` doit pouvoir écrire dans `installers` : c'est là que l'auto-internalisation dépose les installeurs tiers rapatriés.

3.2 Binaire

Télécharger la release depuis GitHub.

Épingler une version précise. Le projet est en pré-1.0 : l'auteur indique que configuration, sortie et comportement peuvent changer à tout moment.

```
VERSION="0.15.0"
cd /tmp
# adapter le nom d'archive à celui de la release
tar xzf rewinged_${VERSION}_linux_amd64.tar.gz
install -m 0755 rewinged /usr/local/bin/rewinged
/usr/local/bin/rewinged --help
```

3.3 Options utiles

Flag	Rôle
`-manifestPath`	Répertoire des manifests (défaut `./packages`)
`-autoInternalize`	Rapatrie et sert localement les installeurs distants
`-autoInternalizePath`	Répertoire des installeurs servis (défaut `./installers`)
`-trustedProxies`	IP du frontal, pour journaliser l'IP réelle du client
`-listen`	Adresse et port d'écoute
`-sourceAuthType`	none ou `microsoftEntraId`
`-https`	TLS assuré par rewinged, inutile derrière un frontal

Flag	Rôle
`-logLevel`	`disable`, `error`, `warn`, `info`, `debug`, `trace`

Deux usages de `-autoInternalizePath`. rewinged y dépose les installeurs qu'il rapatrie automatiquement, et sert directement tout fichier qui s'y trouve sous `/installers/`. Les MSI synchronisés depuis Forgejo y sont donc servis sans serveur de fichiers supplémentaire.

Vérifié :

```
curl -I http://10.0.0.20:8080/installers/mis-agent-1.2.0.msi
# HTTP/1.1 200 OK
# Content-Type: application/x-msi
# Accept-Ranges: bytes
```

3.4 Service systemd

/etc/systemd/system/rewinged.service

```
[Unit]
Description=rewinged winget package source
After=network.target

[Service]
User=rewinged
Group=rewinged
ExecStart=/usr/local/bin/rewinged \
  -listen 10.0.0.20:8080 \
  -manifestPath /srv/winget/manifests \
  -autoInternalize \
  -autoInternalizePath /srv/winget/installers \
  -trustedProxies 10.0.0.10 \
  -logLevel info
Restart=on-failure
RestartSec=5

# Journalisation vers fichier, pour collecte Wazuh en JSON natif
StandardOutput=append:/var/log/rewinged/rewinged.log
StandardError=append:/var/log/rewinged/rewinged.log

# Durcissement
ProtectSystem=strict
ProtectHome=true
PrivateTmp=true
PrivateDevices=true
NoNewPrivileges=true
RestrictSUIDSGID=true
LockPersonality=true
MemoryDenyWriteExecute=true
ReadOnlyPaths=/srv/winget/manifests
```

```
ReadWritePaths=/srv/winget/installers /var/log/rewinged

[Install]
WantedBy=multi-user.target
```

```
mkdir -p /var/log/rewinged
chown rewinged:rewinged /var/log/rewinged

systemctl daemon-reload
systemctl enable --now rewinged
systemctl status rewinged
```

Rotation du journal :

/etc/logrotate.d/rewinged

```
/var/log/rewinged/rewinged.log {
    daily
    rotate 30
    compress
    missingok
    notifempty
    copytruncate
    su rewinged rewinged
}
```

4. Pare-feu de la VM

Seul le frontal doit joindre rewinged.

```
CADDY_IP="10.0.0.10"
ufw default deny incoming
ufw allow from ${CADDY_IP} to any port 8080 proto tcp
ufw allow from 100.64.0.0/10 to any port 22 proto tcp
ufw enable
```

5. Configuration du frontal Caddy

À ajouter au Caddyfile **de la machine frontale**.

/etc/caddy/Caddyfile

```
repo.makeitsimple.be {
    tls vincent@makeitsimple.be
```

```
    reverse_proxy 10.0.0.20:8080

    header {
        Strict-Transport-Security "max-age=31536000"
        X-Content-Type-Options "nosniff"
        -Server
    }

    log {
        output file /var/log/caddy/repo.log
        format json
    }
}
```

Winget exige HTTPS avec une chaîne validée par le magasin de confiance Windows. Let's Encrypt sur un nom public évite d'avoir à déployer une CA interne sur chaque poste.

Un seul `reverse_proxy` : rewinged sert à la fois l'API sous `/api` et les installeurs sous `/installers`.

6. Dépôt Forgejo

Un seul dépôt contient manifests et installeurs. Les MSI passent par Git LFS pour ne pas alourdir l'historique.

6.1 Structure

```
winget-manifests/
├── .gitattributes
├── manifests/
│   └── m/MakeItSimple/
│       ├── Agent/1.2.0/MakeItSimple.Agent.yaml
│       └── RustDesk/1.4.3/MakeItSimple.RustDesk.yaml
└── installers/
    ├── mis-agent-1.2.0.msi
    └── rustdesk-1.4.3.msi
```

6.2 Activation de Git LFS

Côté poste de travail, une seule fois :

```
git lfs install
git lfs track "*.msi"
git lfs track "*.exe"
git add .gitattributes
```

```
git commit -m "Suivi LFS des installeurs"
git push
```

Vérifier que LFS est activé sur l'instance Forgejo et que le quota de stockage est suffisant. LFS ne fait pas de delta : chaque version d'un MSI est stockée intégralement.

6.3 Clone initial sur la VM

```
apt install git-lfs
sudo -u rewinged git lfs install --skip-repo

sudo -u rewinged git clone \
  git@forgejo.makeitsimple.be:mis/winget-manifests.git \
  /srv/winget

sudo -u rewinged git -C /srv/winget lfs pull
```

La clé de déploiement doit être en **lecture seule** : le serveur de paquets ne doit jamais pouvoir écrire dans le dépôt.

7. Synchronisation automatique

Pas de webhook : un timer évite d'exposer un endpoint entrant sur la machine qui décide de ce que le parc installe. Le flux part toujours du serveur vers Forgejo.

7.1 Rechargement natif

rewinged surveille le répertoire de manifests et recharge sans intervention. Confirmé au démarrage du service :

```
{"level":"info","message":"watching manifestPath for changes"}
```

Conséquence : aucun redémarrage n'est nécessaire après une publication. L'unité de synchronisation tourne donc intégralement sous le compte `rewinged`, sans privilège root ni règle polkit.

7.2 Script de synchronisation

/usr/local/bin/winget-sync.sh

```
#!/bin/bash
#
# Synchronise le depot de paquets depuis Forgejo.
# Execute par winget-sync.timer sous le compte rewinged.
#
```

```
set -euo pipefail

REPO_DIR="/srv/winget"
BRANCH="main"

cd "$REPO_DIR"

BEFORE=$(git rev-parse HEAD)

git fetch origin "$BRANCH"
git merge --ff-only "origin/${BRANCH}"
git lfs pull origin

AFTER=$(git rev-parse HEAD)

if [ "$BEFORE" != "$AFTER" ]; then
    # Verification de signature : voir section 13.3
    if ! git verify-commit HEAD >/dev/null 2>&1; then
        logger -t winget-sync -p auth.crit \
            "SIGNATURE INVALIDE sur ${AFTER:0:8} - retour a
${BEFORE:0:8}"
        git reset --hard "$BEFORE"
        exit 1
    fi
    logger -t winget-sync "Publication verifiee: ${BEFORE:0:8} ->
${AFTER:0:8}"
fi

exit 0
```

Le bloc `verify-commit` suppose que les commits sont signés en GPG et que la clé publique du signataire est présente dans le trousseau du compte `rewinged`. Le retirer si la signature n'est pas en place — mais le conserver est ce qui rend la détection Wazuh exploitable (13.3).

```
chown root:root /usr/local/bin/winget-sync.sh
chmod 0755 /usr/local/bin/winget-sync.sh
```

Le script appartient à root en lecture seule pour `rewinged` : le compte de service exécute le script sans pouvoir le modifier.

Le `merge –ff-only` est délibéré : si l'historique local diverge, l'opération échoue au lieu de fusionner silencieusement. Une divergence signifie une écriture locale sur le serveur, ce qui ne doit jamais arriver.

Le `logger` trace chaque publication dans syslog. C'est ce qui permettra de corréler les alertes FIM Wazuh avec une publication légitime (section 12.2).

7.3 Unités systemd

/etc/systemd/system/winget-sync.service

```
[Unit]
Description=Sync winget manifests and installers from Forgejo
After=network-online.target
Wants=network-online.target

[Service]
Type=oneshot
User=rewinged
Group=rewinged
WorkingDirectory=/srv/winget
ExecStart=/usr/local/bin/winget-sync.sh

# Durcissement
ProtectSystem=strict
ProtectHome=true
PrivateTmp=true
NoNewPrivileges=true
ReadWritePaths=/srv/winget
```

/etc/systemd/system/winget-sync.timer

```
[Unit]
Description=Sync winget repository every minute

[Timer]
OnBootSec=1min
OnUnitActiveSec=1min
AccuracySec=10s

[Install]
WantedBy=timers.target
```

```
systemctl daemon-reload
systemctl enable --now winget-sync.timer
systemctl list-timers winget-sync.timer
journalctl -u winget-sync.service -f
```

7.4 Propriété des fichiers

Le clone doit appartenir à `rewinged` de bout en bout. Un clone effectué en root bloque la synchronisation.

```
# Preparation, en root
mkdir -p /srv/winget
chown rewinged:rewinged /srv/winget

# Clone, sous le compte de service
sudo -u rewinged git clone git@forgejo.makeitsimple.be:mis/winget-
manifests.git /srv/winget
```

Rattrapage si le clone a déjà été fait en root :

```
chown -R rewinged:rewinged /srv/winget
sudo -u rewinged git -C /srv/winget status
```

Git refuse d'opérer sur un dépôt dont le propriétaire diffère de l'utilisateur courant (`dubious ownership`). C'est le garde-fou qui signale un `chown` oublié.

Vérification décisive :

```
sudo -u rewinged /usr/local/bin/winget-sync.sh
```

Si elle passe, le timer passera aussi.

7.5 Variante avec redémarrage

À n'utiliser que si le rechargement natif s'avère insuffisant — par exemple après une montée de version de rewinged qui modifierait ce comportement.

Autorisation ciblée dans sudoers :

/etc/sudoers.d/winget-sync

```
rewinged ALL=(root) NOPASSWD: /usr/bin/systemctl try-reload-or-restart
rewinged
```

```
visudo -c -f /etc/sudoers.d/winget-sync
chmod 0440 /etc/sudoers.d/winget-sync
```

Ligne à ajouter dans le bloc conditionnel du script :

```
    sudo /usr/bin/systemctl try-reload-or-restart rewinged
```

Redémarrer uniquement sur changement de HEAD. À raison de 1440 exécutions par jour, un redémarrage systématique pollue les journaux et provoque des micro-coupures pendant les téléchargements.

8. Format d'un manifest

MakeItSimple.Agent.yaml

```yaml
PackageIdentifier: MakeItSimple.Agent
PackageVersion: 1.2.0
PackageLocale: en-US
Publisher: Make IT Simple SRL
PackageName: MIS Agent
License: Proprietary
ShortDescription: Agent d'inventaire et de mise a jour
Installers:
  - Architecture: x64
    InstallerType: msi
    InstallerUrl:
https://repo.makeitsimple.be/installers/mis-agent-1.2.0.msi
    InstallerSha256: A1B2C3D4E5F6...
    Scope: machine
    InstallerSwitches:
      Silent: /quiet /norestart
    AppsAndFeaturesEntries:
      - ProductCode: '{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}'
        UpgradeCode: '{YYYYYYYY-YYYY-YYYY-YYYY-YYYYYYYYYYYY}'
ManifestType: singleton
ManifestVersion: 1.6.0
```

Calcul du hash :

```
sha256sum mis-agent-1.2.0.msi | tr 'a-f' 'A-F'
```

Types d'installeurs supportés : `msi`, `exe`, `inno`, `nullsoft`, `wix`, `burn`, `msix`, `appx`, `zip`, `portable`. Ne pas utiliser `portable` pour un service Windows — il ne fait qu'un alias sur le PATH.

9. Convention pour les paquets par client

RustDesk est publié en un paquet par client : le mot de passe d'accès non-surveillé est préconfiguré dans le MSI et diffère d'un client à l'autre.

9.1 Nommage

```
PackageIdentifier: Rustdesk.<CodeClient>
```

- manifest : `manifests/r/Rustdesk/<CodeClient>/<version>/Rustdesk.<CodeClient>.yaml`
- installeur : `installers/rustdesk-<codeclient>.msi`

9.2 Conséquence opérationnelle

À chaque version de RustDesk, autant de MSI à régénérer et de manifests à republier qu'il y a de clients. À l'échelle de la centaine, la publication doit être scriptée : un outil qui parcourt la liste des clients, appelle `Get-MsiCodes` (section 10.1), calcule le SHA256 et génère le YAML.

La saisie manuelle n'est pas tenable et expose au risque de coller le GUID d'une version précédente.

9.3 Visibilité

La source n'a pas de notion de périmètre : tout poste ayant enregistré `mis-repo` peut énumérer l'ensemble des paquets, donc la liste des codes clients. Sans conséquence sur l'accès aux machines, mais à connaître.

9.4 Alternative : paquet unique

Le paquet par client existe parce que le mot de passe non-surveillé est préconfiguré dans le MSI. Ce mot de passe peut aussi être posé **après** installation, ce qui autorise un paquet unique pour tout le parc.

Ce qui n'est pas possible

Le mot de passe permanent **ne se propage pas par API ni par stratégie**. C'est un secret local à la machine. La demande d'en faire un paramètre de stratégie Pro est ouverte depuis 2023 et n'a pas abouti.

Ne pas confondre trois objets distincts :

Élément	Où il vit	Comment il se pose
Mot de passe permanent	Sur la machine, haché localement	`RustDesk.exe --password` en local
Mot de passe du carnet	Côté serveur Pro	`--assign --address_book_password`
Stratégie Pro	Côté serveur Pro	Réglages client, **hors** mot de passe

Le premier authentifie les connexions entrantes. Le second est celui que le client technicien utilise pour remplir automatiquement le champ.

Le flux en deux temps

Le secret naît sur la machine, puis remonte vers le serveur.

onboard-rustdesk.ps1

```
$rd = "C:\Program Files\RustDesk\RustDesk.exe"

# 1. Generer un mot de passe propre a CETTE machine
```

```
$pw = -join ((48..57)+(65..90)+(97..122) | Get-Random -Count 20 | %
{[char]$_})

# 2. Le poser localement
& $rd --password $pw

# 3. Recuperer l'ID attribue
$id = (& $rd --get-id).Trim()

# 4. Rattacher au carnet du client et y stocker le mot de passe
& $rd --assign `
    --token $env:RUSTDESK_API_TOKEN `
    --address_book_name "Client-$ClientCode" `
    --address_book_alias $env:COMPUTERNAME `
    --address_book_password $pw

# 5. Archiver dans Vaultwarden (ID + mot de passe)
```

Ce que cela change

- **Un seul manifest** RustDesk pour l'ensemble du parc
- **Un mot de passe par machine** au lieu d'un par client : la fuite d'un

secret n'expose plus tout le parc d'un client

- Plus de régénération de MSI à chaque release
- Prérequis : Server Pro 1.5.8 et client 1.4.1 pour `–address_book_alias`

À valider avant de basculer

<WRAP center round important> Un dysfonctionnement de `RustDesk.exe –password` sous Windows a été signalé en mai 2025 : la commande retourne sans effet. Tester sur un poste témoin, service arrêté puis démarré, avant de bâtir l'onboarding dessus. </WRAP>

Vérifier également que le token API utilisé porte les seules permissions `Device` et `Address Book`, et qu'il n'est pas écrit en clair dans le script — le passer par variable d'environnement ou le récupérer depuis Vaultwarden au moment de l'exécution.

10. Extraction des codes MSI

Le `ProductCode` change à chaque version d'un MSI bien construit ; l'`UpgradeCode` reste stable et identifie le produit dans la durée. Le bloc `AppsAndFeaturesEntries` en dépend, notamment pour l'adoption des logiciels déjà installés depuis d'anciens paquets custom.

10.1 Depuis un fichier MSI

À exécuter sur la machine de build (Windows). Lecture seule, sans installation ni Orca.

Get-MsiCodes.ps1

```
function Get-MsiCodes {
    param([string]$Path)
    $installer = New-Object -ComObject WindowsInstaller.Installer
    $db =
$installer.GetType().InvokeMember('OpenDatabase','InvokeMethod',$null,$
installer,@($Path,0))
    $props = @{}
    foreach ($p in
'ProductCode','UpgradeCode','ProductVersion','ProductName') {
        $view =
$db.GetType().InvokeMember('OpenView','InvokeMethod',$null,$db,
                  @("SELECT Value FROM Property WHERE Property='$p'"))
$view.GetType().InvokeMember('Execute','InvokeMethod',$null,$view,$null
)
        $rec =
$view.GetType().InvokeMember('Fetch','InvokeMethod',$null,$view,$null)
        if ($rec) {
            $props[$p] =
$rec.GetType().InvokeMember('StringData','GetProperty',$null,$rec,1)
        }
    }
    [PSCustomObject]$props
}

Get-MsiCodes 'C:\build\mis-agent-1.2.0.msi'
```

Sortie type :

```
ProductName    : MIS Agent
ProductVersion : 1.2.0
ProductCode    : {A1B2C3D4-1111-2222-3333-444455556666}
UpgradeCode    : {99887766-AAAA-BBBB-CCCC-DDDDEEEEFFFF}
```

Le COM `WindowsInstaller.Installer` n'existe que sous Windows : ce script ne tourne pas sur la VM Debian.

10.2 Depuis un poste où le logiciel est installé

Pour recenser les GUID en circulation dans le parc :

```
Get-ItemProperty `
```

```
  HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\*, `
  HKLM:\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall\* |
  Where-Object DisplayName -like '*RustDesk*' |
  Select-Object DisplayName, DisplayVersion, PSChildName
```

`PSChildName` porte le ProductCode lorsque la clé est un GUID, ce qui est le cas des MSI. L'UpgradeCode n'y figure pas.

<WRAP center round important> Éviter `Get-CimInstance Win32_Product` en production : la classe déclenche une réévaluation de tous les MSI installés, ce qui est lent et peut générer des événements de réparation. Acceptable ponctuellement sur un poste témoin. </WRAP>

10.3 Automatisation

L'agent d'inventaire devrait remonter le ProductCode de chaque logiciel installé. Cela donne côté serveur la liste exacte des GUID du parc, donc les `AppsAndFeaturesEntries` à renseigner, sans interroger les postes un par un.

11. Workflow de publication

1. Construire l'installeur
2. Relever les codes avec `Get-MsiCodes` (9.1)
3. Calculer le SHA256
4. Placer le MSI dans `installers/` et le manifest dans `manifests/`
5. Commit sur une branche, ouvrir une PR, revue, merge sur `main`
6. Le timer applique dans la minute
7. Valider depuis un poste témoin avant diffusion

```
winget search -s mis-repo MakeItSimple
winget install --id MakeItSimple.Agent -s mis-repo --silent
```

12. Configuration des postes

12.1 Enregistrement de la source

```
winget source add -n mis-repo `
  -a https://repo.makeitsimple.be/api `
  -t "Microsoft.Rest"
```

À exécuter en **scope machine**, sinon la source n'existe que pour l'utilisateur courant et reste invisible à l'agent tournant en service.

12.2 Winget en contexte SYSTEM

`winget` n'est pas dans le PATH pour le compte SYSTEM. Résoudre le chemin versionné :

```
$winget = (Get-ChildItem `
  "C:\Program
Files\WindowsApps\Microsoft.DesktopAppInstaller_*_x64__8wekyb3d8bbwe\winget.
exe" |
  Select-Object -Last 1).FullName
```

Ajouter `–disable-interactivity` à tous les appels non interactifs, sinon un prompt bloque le processus indéfiniment.

12.3 Adoption du parc existant

Un paquet installé autrement que depuis la source privée n'est pas reconnu par `winget upgrade` tant qu'il n'a pas été installé une fois depuis cette source. Deux contournements :

- renseigner les `AppsAndFeaturesEntries` des anciens builds
- passer une réinstallation depuis `mis-repo`

À valider sur un poste témoin **avant** de dimensionner la campagne.

12.4 Compatibilité

Système	App Installer
Windows 10 1809+ / 11	Livré
Windows Server 2025	Livré
Windows Server 2019 / 2022	**Absent** — installation manuelle, non supportée
Éditions N et LTSC sans Store	Installation manuelle

Inventorier la part de Server 2019/2022 du parc : elle décide si winget est le canal unique ou s'il faut un second canal.

13. Surveillance Wazuh

Trois sources distinctes : l'intégrité des fichiers, le journal applicatif de rewinged, et l'état des services.

13.1 Configuration de l'agent

/var/ossec/etc/ossec.conf

```
<ossec_config>

  <!-- Integrite des fichiers -->
  <syscheck>
```

```
    <disabled>no</disabled>
    <frequency>3600</frequency>

    <!-- Manifests : decident de ce que le parc installe -->
    <directories check_all="yes" realtime="yes" report_changes="yes"
tags="winget,manifests">/srv/winget/manifests</directories>

    <!-- Installeurs : les binaires servis aux postes -->
    <directories check_all="yes" realtime="yes"
tags="winget,installers">/srv/winget/installers</directories>

    <!-- Binaire rewinged et script de synchronisation -->
    <directories check_all="yes" realtime="yes"
                 tags="winget,binaries">/usr/local/bin</directories>

    <!-- Unites systemd du service -->
    <directories check_all="yes" realtime="yes"
tags="winget,systemd">/etc/systemd/system</directories>

    <!-- Bruit git interne -->
    <ignore type="sregex">^/srv/winget/\.git/</ignore>
  </syscheck>

  <!-- Journal applicatif rewinged, deja en JSON -->
  <localfile>
    <log_format>json</log_format>
    <location>/var/log/rewinged/rewinged.log</location>
  </localfile>

  <!-- Etat des services, toutes les 5 minutes -->
  <localfile>
    <log_format>full_command</log_format>
    <command>systemctl is-active rewinged winget-sync.timer | tr '\n' '
'</command>
    <alias>winget-services</alias>
    <frequency>300</frequency>
  </localfile>

</ossec_config>
```

Le journal de `winget-sync` passe par `logger`, donc par syslog, déjà collecté par la configuration Wazuh par défaut.

13.2 Règles personnalisées

/var/ossec/etc/rules/local_rules.xml

```
<group name="winget,rewinged,">
```

```
  <rule id="100600" level="0">
    <decoded_as>json</decoded_as>
    <location>/var/log/rewinged/rewinged.log</location>
    <description>rewinged: evenement applicatif</description>
  </rule>

  <rule id="100601" level="7">
    <if_sid>100600</if_sid>
    <field name="level">error</field>
    <description>rewinged: erreur applicative</description>
  </rule>

  <rule id="100602" level="10">
    <if_sid>100601</if_sid>
    <field name="message">ingestManifests</field>
    <description>rewinged: echec ingestion des manifests -
$(message)</description>
  </rule>

  <rule id="100610" level="3">
    <program_name>winget-sync</program_name>
    <match>Publication verifiee</match>
    <description>winget: publication synchronisee et
verifiee</description>
  </rule>

  <rule id="100611" level="14">
    <program_name>winget-sync</program_name>
    <match>SIGNATURE INVALIDE</match>
    <description>winget: signature invalide - synchronisation
annulee</description>
    <group>attack,</group>
  </rule>

  <rule id="100612" level="10" frequency="5" timeframe="1800">
    <if_matched_sid>100601</if_matched_sid>
    <description>winget: erreurs applicatives repetees</description>
  </rule>

  <rule id="100620" level="3">
    <if_group>syscheck</if_group>
    <field name="file">/srv/winget/manifests/</field>
    <description>winget: manifest modifie - $(file)</description>
  </rule>

  <rule id="100621" level="12">
    <if_group>syscheck</if_group>
    <field name="file">/srv/winget/installers/</field>
    <description>winget: installeur modifie - $(file)</description>
    <group>attack,</group>
```

```
  </rule>

  <rule id="100622" level="14">
    <if_group>syscheck</if_group>
    <field name="file">/usr/local/bin/rewinged</field>
    <description>winget: binaire rewinged modifie -
$(file)</description>
    <group>attack,</group>
  </rule>

  <rule id="100630" level="0">
    <location>winget-services</location>
    <description>winget: releve d'etat des services</description>
  </rule>

  <rule id="100631" level="10">
    <if_sid>100630</if_sid>
    <regex>inactive|failed|unknown</regex>
    <description>winget: service rewinged ou timer arrete</description>
  </rule>

</group>
```

<WRAP center round important> Les identifiants 100600–100631 sont donnés à titre indicatif. Les adapter à la numérotation des règles personnalisées déjà en place. </WRAP>

13.3 Traiter le bruit FIM

La FIM se déclenche à chaque publication légitime. Wazuh ne sait pas exprimer nativement « alerter sauf si précédé de tel événement », il faut donc déplacer le contrôle plutôt que tenter de corréler.

Approche retenue : la signature GPG des commits est le contrôle réel.

1. Les commits de publication sont signés ; la clé publique du signataire est

dans le trousseau du compte `rewinged`

1. `winget-sync.sh` vérifie la signature et annule la synchronisation en cas

d'échec (section 7.2)

1. La FIM sur `manifests` reste **informative** (niveau 3) : elle documente,

elle n'alerte pas

1. L'alerte forte porte sur la signature invalide (100611, niveau 14)

Une modification de manifest effectuée directement sur le serveur ne passe pas par un commit, donc ne déclenche pas 100611 — mais elle sera écrasée au prochain `merge –ff-only`, et la FIM en garde la trace.

Les installeurs et les binaires, eux, ne changent pas hors publication : leurs règles restent à niveau élevé.

13.4 Mise en place de la signature

Côté poste de publication :

```
git config --global user.signingkey <ID_CLE>
git config --global commit.gpgsign true
gpg --export --armor <ID_CLE> > mis-publication.pub
```

Côté serveur rewinged :

```
sudo -u rewinged gpg --import mis-publication.pub
sudo -u rewinged gpg --edit-key <ID_CLE> trust   # niveau ultimate
sudo -u rewinged git -C /srv/winget verify-commit HEAD
```

Idéalement la clé de signature réside sur une YubiKey, cohérent avec l'autorité de certification SSH déjà en place.

13.5 Ce qui reste non couvert

- **Publication légitime mais erronée** — un mauvais `PackageIdentifier`

ou un SHA256 incorrect signé de bonne foi passe toutes les vérifications.

```
  Le garde-fou est la revue de la PR, pas la détection.
* **Compromission du poste de publication** — si la clé de signature est
  utilisable par un attaquant, la chaîne entière est valide de son point de
  vue. D'où la YubiKey.
* **Saturation disque** — à surveiller côté Zabbix plutôt que Wazuh, la
  croissance LFS étant progressive et prévisible.
```

14. Points ouverts

Sujet	Action
Auto-internalisation	Valider le rapatriement des installeurs tiers depuis les URL éditeur
Bug de matching	Tester sur un poste témoin avec un logiciel préinstallé
Server 2019/2022	Inventorier la proportion dans le parc
Upgrade RustDesk	Vérifier que la mise à jour préserve `RustDesk2.toml` et le mot de passe permanent
`–password` sous Windows	Tester le réglage du mot de passe en ligne de commande (9.4) avant d'envisager le paquet unique
Rétention LFS	Définir une politique de purge des anciennes versions d'installeurs
Quota Forgejo	Dimensionner le stockage LFS et surveiller sa croissance

15. Références

- rewinged — dépôt GitHub
- Documentation Windows Package Manager
- RustDesk Server Pro — console et tokens API

From:
https://wiki.makeitsimple.be/ - **makeITsimple wiki**

Permanent link:
https://wiki.makeitsimple.be/doku.php?id=linux:rewinged

Last update: **2026/09/18 17:40**