Les problèmes de configuration qui empêchent la mise en production

Sommaire

Contexte

Cette page a pour but de décrire la mise en place d'une configuration minimale nécessaire pour un hôte supervisé par le pack **windows-by-WinRM__shinken.**

Procédure de configuration

La supervision d'un hôte par un **Poller Shinken** se fait par requête WinRM.

- Sur le **Poller shinken**, la sonde SNMP installé est responsable d'envoyer les requêtes et de traiter les réponses.
- Sur les hôtes à superviser, le service windows WinRM qui est responsable de répondre aux requêtes.

Le service Windows **WinRM** sur l'hôte supervisée permet l'exécution de commande à distance, et notamment la récupération d'informations sur la machine mise à disposition par **WMI** (Windows Management Instrumentations).

Alors, il est nécessaire de configurer le service **WinRM**, un utilisateur qui exécutera des commandes via **WinRM** et accèdera aux informations nécessaires aux checks.

> Les commandes ci-dessous doivent être exécutées par un administrateur local de la machine (*sauf mention contraire*).

Configuration de WinRM

Le service **WinRM** est installé par défaut sur les systèmes d'exploitations à partir de Windows 7 / Windows Server 2008 R2, mais celui-ci n'est pas nécessairement démarré et configuré.

Voici la commande à exécuter via PowerShell pour vérifier si le service est bien démarré :

```
Get-Service WinRM
```

Exemple :

```
C:\Users\Administrateur> Get-Service WinRM

Status   Name               DisplayName
------   ----               -----------
Running  WinRM              Gestion à distance de Windows (Gest...
```

Configuration minimale

WinRM dispose d'une commande de configuration intégrée qui permet entre autres de :

- Démarrer le service
- Activer le démarrage automatique de celui-ci
- Mettre en place l'écouteur (HTTP ou HTTPS) pour recevoir les requêtes des sondes Shinken
- Configurer le Firewall Windows pour autoriser les accès vers l'écouteur

Pour effectuer les actions de configuration, il suffit d'exécuter la commande suivante :

```
winrm quickconfig
```

Exemple :

```
C:\Users\Administrateur> winrm quickconfig

WinRM n'est pas configuré pour la gestion à distance de cet ordinateur.
Les modifications suivantes doivent être effectuées :

Créez un écouteur WinRM sur HTTP://* pour accepter les demandes de la gestion des services Web sur toutes
les adresses IP de cet ordinateur.

Activez l'exception de pare-feu WinRM.
Configurez LocalAccountTokenFilterPolicy pour attribuer des droits d'administration à distance à des
utilisateurs locaux.

Effectuer ces modifications [y/n] ? y

WinRM a été mis à jour pour la gestion à distance.

Écouteur WinRM créé sur HTTP://* pour accepter les demandes de la gestion des services Web sur toutes les
adresses IP de cet ordinateur.

Exception de pare-feu WinRM activée.
LocalAccountTokenFilterPolicy configuré pour attribuer des droits d'administration à distance à des
utilisateurs locaux.
```

Authentification

Selon le choix du mode d'authentification, il faut configurer l'authentification à WinRM.

Par défault le service WinRM est configuré pour autoriser l'authentification "Negotiate", mais désactive par défaut "Basic".

> L'utilisation du protocole « Basic » n'est pas recommandée lorsque des modes d'authentification plus sécurisés sont disponibles sur le système.

Pour activer le mode d'authentification souhaité, utiliser l'une des commandes suivantes.

```
winrm set winrm/config/service/auth '@{Negotiate="true"}'

winrm set winrm/config/service/auth '@{Basic="true"}'
```

Ensuite il est nécessaire d'activer les connexions non chiffrées.

```
winrm set winrm/config/service '@{AllowUnencrypted="true"}'
```

La sonde implémentera prochainement le support HTTPS afin d'activer le chiffrement des messages.

Néanmoins, les messages d'authentifications de "Negotiate" bénéficient d'un chiffrement et n'exposent jamais les mots de passe en clair, contrairement au mode "Basic".

Configuration de l'utilisateur

À l'image des commandes WMI (*voir la page Modèle windows-by-WMI__ntlmv2*), les commandes WinRM requièrent une authentification au préalable afin de récupérer les informations de supervision sur l'hôte windows. (*-u "$_HOSTDOMAINUSER$" -p "$_HOSTDOMAINPASSWORD$"*)

L'utilisation du compte administrateur du poste Windows permet d'obtenir toutes les informations du système, car celui-ci possède par défaut tous les droits d'accès (*WMI, DCOM, WinRM, etc.*).

Cependant, pour des raisons de sécurité, il n'est pas conseillé de l'utiliser pour effectuer la supervision. Il faut alors créer un utilisateur qui exécutera les commandes demandées par les sondes, avec uniquement les droits nécessaires.

Qu'il s'agisse d'un compte local ou d'un compte AD, voici la procédure pour créer un nouvel utilisateur dédié au monitoring avec les droits suffisant pour collecter les informations nécessaires au fonctionnement des sondes fournies par Shinken.

Création de l'utilisateur

La création est possible directement sur le poste local Windows, ou sur l'Active Directory.

Par interface

Cela s'effectue dans la console de "**Gestion de l'ordinateur**" (*compmgmt.msc*) puis dans **Utilisateurs et groupes locaux > Utilisateurs,** clic droit et **Nouvel utilisateur...**

? Unknown Attachment

Par ligne de commande

Dans un PowerShell :

```
net user "shinken" "mon_motdepasse" /ADD
```

Remplacer ici "mon_motdepasse".

Configuration de la langue

Pour assurer l'interprétation des commandes Windows par la sonde, il est nécessaire de configurer la langue du nouvel utilisateur.

Il est nécessaire de se connecter au nouveau compte créer pour changer la langue de l'utilisateur.

Par interface

Lancer les paramètres Windows puis accéder à la catégorie "Time & Language".

? Unknown Attachment

Ensuite accéder à la sous-catégorie "**Language**".

Maintenant, dans la section "**Windows display language**", sélectionner "**English (United States)**".

Ensuite, dans la section "**Preferred languages**", ajouter la lange "**English (United States)**". Déplacer là en première position de la liste.

? Unknown Attachment

Par ligne de commande

Après avoir ouvert un PowerShell, exécuter les commande suivante :

```
Install-Language -Language en-US
Set-WinUserLanguageList en-US -Force
```

Ensuite, il est nécessaire de **se déconnecter puis se reconnecter** au nouvel utilisateur afin de correctement appliquer le changement de langue.

Une fois l'opération réalisée, il est possible de se reconnecter au compte administrateur Windows et de poursuivre la configuration.

Configuration des groupes

Par interface

Une fois le nouvel utilisateur créé sur le poste client ou sur le domaine, et sa langue configurée, il faut ajouter l'utilisateur dans les groupes suivants : **U tilisateurs de gestion à distance** et **Utilisateurs de l'Analyseur de performances** sur le poste Windows concerné.

Cela s'effectue dans la console de "**Gestion de l'ordinateur**" (*compmgmt.msc*) puis dans **Utilisateurs et groupes locaux > Groupes,** clic droit sur le groupe puis **Propriétés**. Une nouvelle fenêtre s'ouvre, cliquer sur **Ajouter…** :

> Sur certaines distributions inférieures à Windows 2012, le groupe "**Utilisateurs de gestion à distance**" et sa fonction n'existent pas.

? Unknown Attachment

Par ligne de commande

Dans un PowerShell :

- Si la langue de l'utilisateur administrateur est anglais :

```
net localgroup "Remote Management Users" "shinken" /ADD
net localgroup "Performance Monitor Users" "shinken" /ADD
```

- Si la langue de l'utilisateur administrateur est français :

```
net localgroup "Utilisateurs de gestion à distance" "shinken" /ADD
net localgroup "Utilisateurs de l’Analyseur de performances" "shinken" /ADD
```

Il est possible que la commande "net localgroup "Utilisateurs de l’Analyseur de performances" "shinken" /ADD" ne fonctionne pas lors d'un copier-coller.

Il y a un bug avec le terminal PowerShell sur le copier-coller qui ne retranscrira pas le symbole apostrophe ’ lorsque l'action "coller" est déclenché avec un clic droit dans le terminal.
Il faut utiliser le raccourci "CTRL+V" affin de coller correctement la chaine.
Sinon, il est possible de générer le caractère spécial apostrophe ’ avec le raccourci "ALT + 0146". (*Différent de l'apostrophe de la touche 4*).

Example du bug du copier-coller dans un terminal PowerShell Administrateur où l'apostrophe n'est pas retranscrit :

? Unknown Attachment

.

Configuration des permissions

Permissions WinRM pour l'utilisateur

Il est nécessaire d'ajouter les droits de lecture et d'exécution aux commandes WinRM au nouvel utilisateur :

Par interface

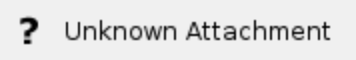
Dans une console PowerShell, exécuter la commande suivante :

```
winrm configSDDL default
```

Une nouvelle fenêtre s'ouvre. Dans celle-ci, ajouter le nouvel utilisateur et cliquer sur **Ajouter...** et attribuer les droits :

- Lecture(Get,Enumerate,Subscribe)
- Exécution(Invoke)

En cochant les cases correspondantes dans le tableau des droits situé en dessous de la liste des utilisateurs (*cliquer sur l'utilisateur au préalable*).



Par ligne de commande

Dans un PowerShell :

```
$user = "shinken"

$GENERIC_READ = 0x80000000
$GENERIC_EXECUTE = 0x20000000

$user_sid = (New-Object -TypeName System.Security.Principal.NTAccount -ArgumentList $user).Translate([System.
Security.Principal.SecurityIdentifier])

# get the existing SDDL of the WinRM listener
$sddl = (Get-Item -Path WSMan:\localhost\Service\RootSDDL).Value

# convert the SDDL string to a SecurityDescriptor object
$sd = New-Object -TypeName System.Security.AccessControl.CommonSecurityDescriptor -ArgumentList $false,
$false, $sddl

# apply a new DACL to the SecurityDescriptor object
$sd.DiscretionaryAcl.AddAccess(
    [System.Security.AccessControl.AccessControlType]::Allow,
    $user_sid,
    ($GENERIC_READ -bor $GENERIC_EXECUTE),
    [System.Security.AccessControl.InheritanceFlags]::None,
    [System.Security.AccessControl.PropagationFlags]::None
)

# get the SDDL string from the changed SecurityDescriptor object
$new_sddl = $sd.GetSddlForm([System.Security.AccessControl.AccessControlSections]::All)

# apply the new SDDL to the WinRM listener
Set-Item -Path WSMan:\localhost\Service\RootSDDL -Value $new_sddl -Force
```

Autorisation aux objets CIM

Par interface

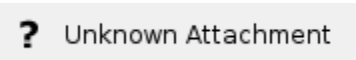
La sonde va demander les informations systèmes via les objets CIM, il est nécessaire d'ajouter les droits à l'utilisateur.

> Cette section n'est pas nécessaire si les machines supervisées sont configurées avec un Active Directory.

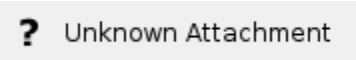
Il faut en premier temps lancer la fenêtre de contrôle WMI avec la commande :

```
wmimgmt.msc
```

Une fois lancé, clic droit sur **Contrôle WMI (local)** puis sélectionner **Propriétés**.



Accéder à la section Sécurité, puis dans l'arborescence ci-dessous, sélectionner **Root > CIMV2** puis cliquer sur le bouton **Sécurité** situé en bas à droite.



Enfin, ajouter l'utilisateur afin de lui appliquer de nouveaux droits, puis cocher **Activer le compte** et **Appel à distance autorisé**.

? Unknown Attachment

Ensuite, répétez l'opération pour StandardCimV2

? Unknown Attachment

Par ligne de commande

Dans un PowerShell. Cette commande va écrire un script PowerShell qu'il permet d'ajouter les droits nécessaires à la lecture des objets CIM.

```
$script_name = ".\Set-WmiNamespaceSecurity.ps1"

@'
# Set-WmiNamespaceSecurity.ps1
# Example: Set-WmiNamespaceSecurity root/cimv2 add steve Enable,RemoteAccess

Param ( [parameter(Mandatory=$true,Position=0)][string] $namespace,
    [parameter(Mandatory=$true,Position=1)][string] $operation,
    [parameter(Mandatory=$true,Position=2)][string] $account,
    [parameter(Position=3)][string[]] $permissions = $null,
    [bool] $allowInherit = $false,
    [bool] $deny = $false,
    [string] $computerName = ".",
    [System.Management.Automation.PSCredential] $credential = $null)

Process {
    $ErrorActionPreference = "Stop"
    Function Get-AccessMaskFromPermission($permissions) {
        $WBEM_ENABLE = 1
        $WBEM_METHOD_EXECUTE = 2
        $WBEM_FULL_WRITE_REP = 4
        $WBEM_PARTIAL_WRITE_REP = 8
        $WBEM_WRITE_PROVIDER = 0x10
        $WBEM_REMOTE_ACCESS = 0x20
        $WBEM_RIGHT_SUBSCRIBE = 0x40
        $WBEM_RIGHT_PUBLISH = 0x80
        $READ_CONTROL = 0x20000
        $WRITE_DAC = 0x40000

        $WBEM_RIGHTS_FLAGS = $WBEM_ENABLE,$WBEM_METHOD_EXECUTE,$WBEM_FULL_WRITE_REP,`

            $WBEM_PARTIAL_WRITE_REP,$WBEM_WRITE_PROVIDER,$WBEM_REMOTE_ACCESS,`

            $READ_CONTROL,$WRITE_DAC

        $WBEM_RIGHTS_STRINGS = "Enable","MethodExecute","FullWrite","PartialWrite",`
            "ProviderWrite","RemoteAccess","ReadSecurity","WriteSecurity"

        $permissionTable = @{}
        for ($i = 0; $i -lt $WBEM_RIGHTS_FLAGS.Length; $i++) {
            $permissionTable.Add($WBEM_RIGHTS_STRINGS[$i].ToLower(), $WBEM_RIGHTS_FLAGS[$i])
        }

        $accessMask = 0
        foreach ($permission in $permissions) {
            if (-not $permissionTable.ContainsKey($permission.ToLower())) {
                throw "Unknown permission: $permission`nValid permissions: $($permissionTable.Keys)"
            }
            $accessMask += $permissionTable[$permission.ToLower()]
        }
        $accessMask
    }

    if ($PSBoundParameters.ContainsKey("Credential")) {
        $remoteparams = @{ComputerName=$computer;Credential=$credential}
    } else {
        $remoteparams = @{}
    }
```

```
$invokeparams = @{Namespace=$namespace;Path="__systemsecurity=@"} + $remoteParams
$output = Invoke-WmiMethod @invokeparams -Name GetSecurityDescriptor
if ($output.ReturnValue -ne 0) {
    throw "GetSecurityDescriptor failed: $($output.ReturnValue)"
}

$acl = $output.Descriptor
$OBJECT_INHERIT_ACE_FLAG = 0x1
$CONTAINER_INHERIT_ACE_FLAG = 0x2
$computerName = (Get-WmiObject @remoteparams Win32_ComputerSystem).Name

if ($account.Contains('\')) {
    $domainaccount = $account.Split('\')
    $domain = $domainaccount[0]
    if (($domain -eq ".") -or ($domain -eq "BUILTIN")) {
        $domain = $computerName
    }
    $accountname = $domainaccount[1]
} elseif ($account.Contains('@')) {
    $domainaccount = $account.Split('@')
    $domain = $domainaccount[1].Split('.')[0]
    $accountname = $domainaccount[0]
} else {
    $domain = $computerName
    $accountname = $account
}

$getparams = @{Class="Win32_Account";Filter="Domain='$domain' and Name='$accountname'"} + $remoteParams
$win32account = Get-WmiObject @getparams
if ($win32account -eq $null) {
    throw "Account was not found: $account"
}

switch ($operation) {
    "add" {
        if ($permissions -eq $null) {
            throw "-Permissions must be specified for an add operation"
        }

        $accessMask = Get-AccessMaskFromPermission($permissions)
        $ace = (New-Object System.Management.ManagementClass("win32_Ace")).CreateInstance()
        $ace.AccessMask = $accessMask

        if ($allowInherit) {
            $ace.AceFlags = $OBJECT_INHERIT_ACE_FLAG + $CONTAINER_INHERIT_ACE_FLAG
        } else {
            $ace.AceFlags = 0
        }

        $trustee = (New-Object System.Management.ManagementClass("win32_Trustee")).CreateInstance()
        $trustee.SidString = $win32account.Sid
        $ace.Trustee = $trustee

        $ACCESS_ALLOWED_ACE_TYPE = 0x0
        $ACCESS_DENIED_ACE_TYPE = 0x1

        if ($deny) {
            $ace.AceType = $ACCESS_DENIED_ACE_TYPE
        } else {
            $ace.AceType = $ACCESS_ALLOWED_ACE_TYPE
        }
        $acl.DACL += $ace.psobject.immediateBaseObject
    }
    "delete" {
        if ($permissions -ne $null) {
           throw "Permissions cannot be specified for a delete operation"
        }

        [System.Management.ManagementBaseObject[]]$newDACL = @()
        foreach ($ace in $acl.DACL) {
```

```
                if ($ace.Trustee.SidString -ne $win32account.Sid) {
                    $newDACL += $ace.psobject.immediateBaseObject
                }
            }
            $acl.DACL = $newDACL.psobject.immediateBaseObject
        }
        default {
            throw "Unknown operation: $operation`nAllowed operations: add delete"
        }
    }

    $setparams = @{Name="SetSecurityDescriptor";ArgumentList=$acl.psobject.immediateBaseObject} +
$invokeParams
    $output = Invoke-WmiMethod @setparams
    if ($output.ReturnValue -ne 0) {
        throw "SetSecurityDescriptor failed: $($output.ReturnValue)"
    }

}


'@ | Set-Content $script_name
```

Ensuite, executer le script deux fois avec les paramètres suivants :

```
$user = "shinken"
.\Set-WmiNamespaceSecurity.ps1 "root\cimv2"  add $user Enable,RemoteAccess
.\Set-WmiNamespaceSecurity.ps1 "root\standardcimv2" add $user Enable,RemoteAccess
```

Il est possible par la suite de supprimer le script :

```
rm .\Set-WmiNamespaceSecurity.ps1
```

Autorisation au service W32Time

Afin de récupérer les informations de temps, il est nécessaire au nouvel utilisateur d'avoir certains accès en lecture au service de temps de windows W32Time.

Par ligne de commande

Dans un PowerShell :

```
$user = "shinken"

$sid = (New-Object System.Security.Principal.NTAccount($user)).Translate([System.Security.Principal.
SecurityIdentifier]).Value
$currentSddl = & sc.exe sdshow w32time
# Ajoute les droits suivants à l'utilisateur
# CC – SERVICE_QUERY_CONFIG
# LC – SERVICE_QUERY_STATUS
# LO – SERVICE_INTERROGATE
$newAce = "(A;;CCLCLO;;;${sid})"
$newSddl = $currentSddl + $newAce

sc.exe sdset w32time "$newSddl"

Restart-Service w32time
```

Appliquer les permissions

Les nouvelles permissions attribuées au nouvel utilisateur seront appliquées lorsque de nouveaux tokens de connections seront générés. Il est possible de :

- Attendre quelques minutes que les anciens tokens expirent et que de nouveaux, ayant de nouvelles permissions, soient générés
- Redémarrer la machine