Gestion de l'authentification NagVis

Sommaire

Problématiques rencontrées avec l'authentification NagVis

Lorsqu'un utilisateur utilise une instance de NagVis, il utilise des identifiants propres à cette installation de NagVis. Il est possible de gérer les utilisateurs et leur droit directement dans NagVis.

Lorsque NagVis est installé pour être utilisé de manière transparente avec Shinken Entreprise, cette fonctionnalité devient un problème puisqu'il devient nécessaire de synchroniser les bases d'utilisateurs de Shinken et de NagVis. Dans Shinken, une base d'utilisateurs est déjà présente.

Pour simplifier la gestion de l'authentification entre NagVis et Shinken, plusieurs modules ont été ajoutés dans NagVis.

Fonctionnement de l'authentification de NagVis

Dans une installation NagVis classique, la gestion des utilisateurs est gérée avec 3 types de modules différents:

- **Un module de login**
  Définit la manière avec laquelle l'utilisateur fournit ses identifiants de connexion. Selon les modules, les identifiants peuvent être passés par un formulaire ou par variable d'environnement.

- **Un module d'authentification**
  Utilise les identifiants de connexion fournis par le module de login et vérifie si ces identifiants sont corrects. Le module d'authentification par défaut vérifie les identifiants de connexion dans la base d'utilisateur propre à NagVis.

- **Un module d'autorisation**
  Utilise les données de l'utilisateur (son profil et ses réglages) pour lui attribuer les droits nécessaires (droits d'administration de NagVis, droits et vue et d'édition des cartes, etc...)

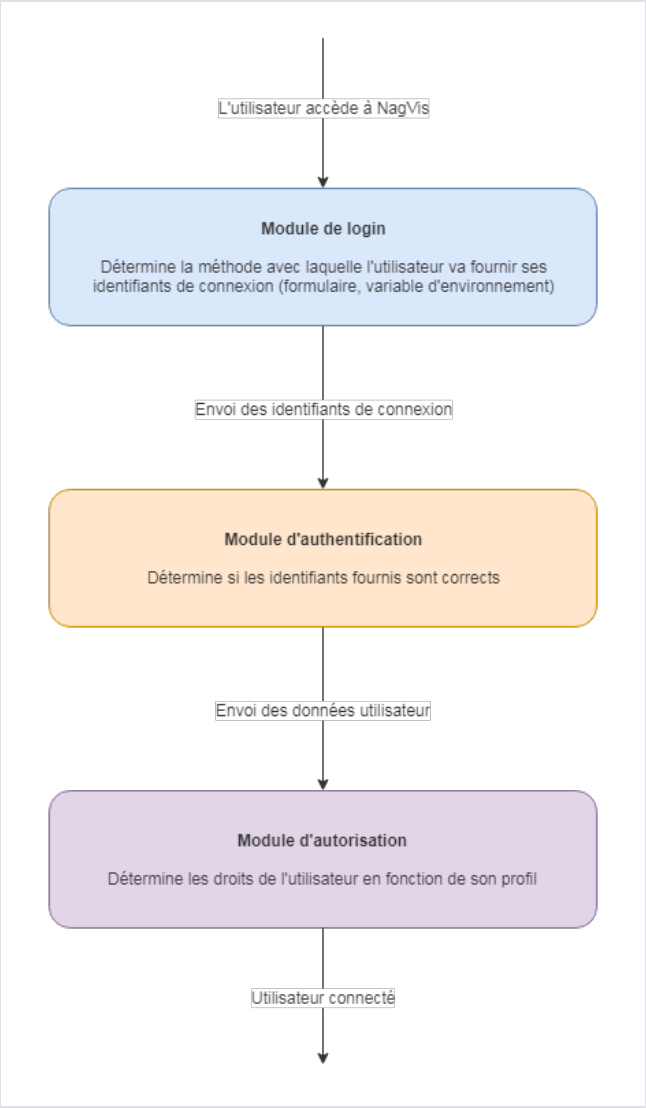
Les modules par défaut dans l'installation NagVis utilisée pour l'export de l'architecture sont les suivants:

- **Module de login**: LogonShinkenMixed
  Récupère les informations de connexion via des entêtes HTTP. Si aucun entête d'authentification n'est passé, les identifiants de connexion sont récupérés depuis le cookie des interfaces Web Shinken. Sinon, un formulaire de connexion classique est

utilisé.

- **Module d'authentification**: CoreAuthModShinken
  Vérifie la validité des identifiants de connexion avec les informations stockées dans la base d'utilisateurs de Shinken.
- **Module d'autorisation**: CoreAuthorisationModShinken
  Définit des droits par défaut (non modifiables).

Le fonctionnement de ces modules est décrit de manière détaillée dans les sections suivantes.



Pour plus d'informations sur les modules disponibles par défaut, la documentation NagVis présente un récapitulatif des fonctionnalités disponibles:

- http://docs.nagvis.org/1.9/en_US/index.html

Solutions d'authentification mises en place

Pour permettre une gestion de l'authentification transparente entre Shinken et NagVis, plusieurs modules ont été ajoutés.

Configuration générale

Pour permettre la liaison de l'authentification avec Shinken, les différents modules utilisés pour la connexion ont besoin de savoir quelle est l'adresse de l'installation Shinken avec laquelle il lui faut se connecter.

De manière plus précise, pour se connecter avec Shinken, NagVis utilise le module WebUI (l'interface de visualisation).

Plusieurs paramètres sont ajoutés pour spécifier l'installation Shinken à contacter:

Option	Type	Valeur par défaut	Commentaire
`authmodule`	Texte	**CoreAuthModShinken**	Module pour vérifier que les utilisateurs fournis sont bien dans la base Shinken. (*Voir la page* Gestion de l'authentification).
`authorisationmodule`	Texte	**CoreAuthorisationModShinken**	Module de gestion des autorisations des utilisateurs. (*Voir la page* Gestion de l'authentification).

`logonmodule`	Texte	**LogonShinkenMixed**	Module de connexion. (*Voir la page* Gestion de l'authentification).
`shinken_auth_restrict_to_shinken_admin`	Booléen	**1**	Restreint la connexion à NagVis aux administrateurs shinken. (*Voir la page* Gestion de l'authentification).
`shinken_auth_protocol`	Texte	**http**	Cette valeur est automatiquement renseignée par le module **architecture-export** de l'Arbiter. Précise le protocole à utiliser pour contacter la WebUI. Les valeurs possibles sont les suivantes : • **http** : pour une connexion non chiffrée • **https** : pour une connexion chiffrée
`shinken_auth_port`	Entier	**7767**	Cette valeur est automatiquement renseignée par le module **architecture-export** de l'Arbiter. Précise le port réseau à utiliser pour contacter la WebUI (*Voir la page* Module architecture-export).
`shinken_auth_address`	Texte	***vide***	Cette valeur est automatiquement renseignée par le module **architecture-export** de l'Arbiter. Précise le nom d'hôte à utiliser pour se connecter à la WebUI (*Voir la page* Module architecture-export).
`shinken_auth_remote_user_variable`	Texte	***vide***	Cette valeur est automatiquement renseignée par le module **architecture-export** de l'Arbiter. Précise le nom de la variable à rechercher dans les entêtes HTTP pour activer l'identification automatique lorsqu'on arrive de la WebUI (*Voir la page* Module architecture-export).
`shinken__authentication__ssl__verify_certificate`	Booléen	**0**	Activer la vérification du certificat reçu de la WebUI quand elle est configurée en **https** • **0** : non • **1** : oui
`shinken__authentication__ssl__verify_certificate_name`	Booléen	**1**	Quand la vérification du certificat de la WebUI est activé, vérifier si le nom d'hôte de la WebUI correspond au nom enregistré dans le ceritifcat • **0** : non • **1** : oui Sous CentOS 7 (*ayant une version de PHP < 7*), la vérification du nom du certificat ne fonctionne pas quand ce certificat est dans la chaine de confiance (*paramètre shinken__authentication__ssl__certificate_authority_file*). Il faut mettre la version de PHP à jour en version 7.2 si cette fonctionnalité est nécessaire.
shinken__authentication__ssl__allow_self_signed_certificate	Booléen	**1**	Quand la vérification du certificat de la WebUI est activé, autoriser les certificats auto-signés • **0** : non • **1** : oui
shinken__authentication__ssl__certificate_authority_file	Texte	***vide***	Définit le certificat d'autorité à utiliser ▪ la valeur par défaut est la chaîne de confiance du système ▪ pour autoriser un certificat auto signé généré pour Shinken, on peut utiliser "/etc/shinken/certs/ca.pem"

Ces paramètres peuvent être modifiés de 2 manières différentes:

- **Par le fichier de configuration de NagVis**:

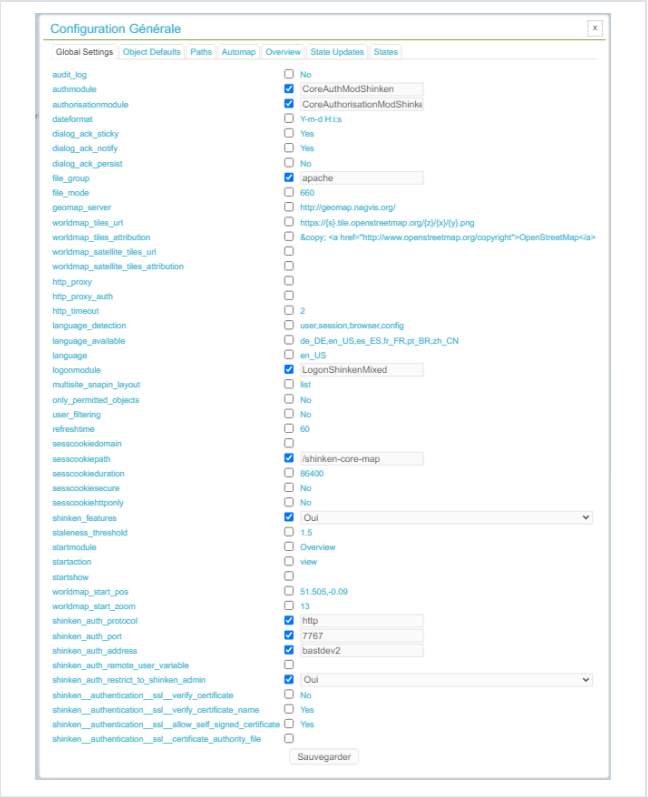
/opt/nagvis/etc/nagvis.ini.php

```
; Defines the authentication module to use. By default NagVis uses the built-in
; SQLite authentication module. On delivery there is no other authentication
; module available. It is possible to add own authentication modules for
; supporting other authorisation mechanisms. For details take a look at the
; documentation.
authmodule="CoreAuthModShinken"

; - CoreAuthorisationModMySQL: Uses the same data structure as the SQLite authorisation
;   module, but stores the data in a MySQL database.
; - CoreAuthorisationModMultisite: Uses information exported by Check_MKs Multisite
;   to gather user permissions. This makes use of the roles defined for a user within
;   multisite and the resulting permissions.
; - CoreAuthorisationModGroups: Assumes all users which should access NagVis are
;   available in your monitoring core as contacts and assigned to contactgroups. Those
;   contact group memberships are matched against a mapping table, which is defined in
;   nagvis/etc/perms.db. This mapping table defines the permissions of each contact
;   group within NagVis. Take a look at the docs for details.
;
; It is possible to add own authorisation modules for supporting other authorisation
; mechanisms. For details take a look at the documentation.
authorisationmodule="CoreAuthorisationModShinkenGroups"

; Protocol to use when authenticating with Shinken (http or https) when using the CoreAuthModShinken
authentication module
;shinken_auth_protocol="http"
; Port of broker webui
;shinken_auth_port=7767
; Address of broker webui. If not specified, address of default backend is used instead
;shinken_auth_address=""
; Name of the HTTP header to use to perform SSO authentication with Shinken.
; This value must be the same as the one configured in Shinken. An empty value means authentication by http
header is disabled.
;shinken_auth_remote_user_variable=""
; When shinken_auth_restrict_to_shinken_admin is set to 1, authentication will be refused if the user is not
a Shinken administrator
;shinken_auth_restrict_to_shinken_admin="0"
; Enable verification of SSL certificate issued by WebUI
;shinken__authentication__ssl__verify_certificate  = 0
; Enable verification of peer name in SSL certificate issued by WebUI
;shinken__authentication__ssl__verify_certificate_name = 1
; Allow WebUI to issue a self signed certificate
;shinken__authentication__ssl__allow_self_signed_certificate = 1
; Set location of certificate authority on local filesystem ( cf. https://www.php.net/manual/en/context.ssl.
php#context.ssl.cafile )
; - system defaults to:
;   - on RedHat / CentOS / Alma / Rocky : "/etc/ssl/certs/ca-bundle.trust.crt"
;   - on Debian : "/etc/ssl/certs/ca-certificates.crt"
; - you can use "/etc/shinken/certs/ca.pem" if you generated a self signed certificate for your Shinken
installation
;shinken__authentication__ssl__certificate_authority_file = ""

  ...

; LogonMixed: The mixed logon module uses the LogonEnv module as default and
;   the LogonDialog module as fallback when LogonEnv returns no username. This
;   should fit the requirements of most environments.
;

...

logonmodule="LogonShinkenMixed"
```

- **Par l'interface Web de NagVis** (Options > Configuration générale).

Le lien vers l'interface Web de NagVis est présent dans l'interface de visualisation (menu "Architectures"), ou directement à l'adresse suivante:

- http://IP-ARBITER/shinken-map/



Modules de login

Utilisation d'entêtes HTTP

Nom du module: CoreLogonShinkenHeader

Lorsque la connexion par SSO est activée dans Shinken (*voir la page* Synchronizer - Authentification unique (SSO)), il est alors possible d'utiliser cette fonctionnalité pour se connecter dans NagVis en utilisant les mêmes entêtes HTTP que ceux utilisés pour Shinken.

L'entête utilisé dans NagVis doit être le même que celui utilisé dans l'interface de Visualisation de Shinken. Pour utiliser l'authentification par SSO dans NagVis avec le module fourni par Shinken, l'authentification par entête HTTP doit également être activée dans Shinken sur l'UI de Visualisation.

Le nom de l'entête contenant le nom d'utilisateur doit être spécifié dans NagVis avec le paramètre "*shinken_auth_remote_user_variable*".

/opt/nagvis/etc/nagvis.ini.php

```
; Name of the HTTP header to use to perform SSO authentication with Shinken.
; This value must be the same as the one configured in Shinken. An empty value means authentication by http
header is disabled.
;shinken_auth_remote_user_variable=""
```

Lorsque cette variable est vide, l'authentification par entête HTTP est désactivée. Par défaut, l'utilisation des entêtes HTTP est donc désactivée dans NagVis.

Utilisation des cookies des interfaces Web Shinken

Nom du module: CoreLogonShinkenCookie

Ce module utilise les cookies des interfaces Web de Shinken Entreprise. Lorsque l'utilisateur est connecté dans une interface (interface de Configuration ou interface de Visualisation), il sera donc automatiquement connecté dans NagVis. La connexion peut être restreinte aux administrateurs Shinken grâce au paramètre "*shinken_auth_restrict_to_shinken_admin*".

⚠ Cette solution ne sera fonctionnelle qui si NagVis est installé sur une machine qui héberge également au moins une interface Shinken (Configuration ou Visualisation). En d'autres mots, cette solution ne sera fonctionnelle qui si le Synchronizer ou un Broker avec le module "webui" sont présents sur la même machine que le démon Arbiter.

Formulaire de connexion

Nom du module: CoreLogonDialog

Ce module est le module par défaut de connexion NagVis. Il affiche un formulaire qui demande à l'utilisateur d'entrer son nom d'utilisateur et son mot de passe pour se connecter dans NagVis.

Lorsqu'il est utilisé avec le module d'authentification "*CoreAuthModShinken*", les identifiants entrés dans le formulaire seront vérifiés avec Shinken.

Agrégation des modules précédents

Nom du module: CoreLogonShinkenMixed *(par défaut)*

Ce module rassemble les différents modes de connexion précédents en un seul module. Le fonctionnement du module est le suivant:

- Tentative de connexion en utilisant les entêtes HTTP
- Si l'authentification avec entête HTTP échoue (*utilisateur invalide, entête HTTP non défini*), le module tente de connecter l'utilisateur en utilisant le cookie des interfaces Web Shinken.
- Si l'authentification par cookie a échoué, le formulaire de connexion est présenté à l'utilisateur.

Modules d'authentification

Authentification avec Shinken Entreprise

Nom du module: CoreAuthModShinken *(par défaut)*

Ce module utilise les données de connexion fournies par le module de login et vérifie auprès de Shinken (*en particulier l'interface de visualisation*) si les identifiants correspondent bien à un utilisateur Shinken existant.

Le comportement de ce module est configurable avec les variables définies dans la section précédente sur la configuration générale des modules d'authentification.

Modules d'autorisation

Définition des droits selon le profil Shinken

Nom du module: CoreAuthorisationModShinken *(par défaut)*

Ce module définit des droits par défaut pour un utilisateur connecté avec Shinken. Ces droits, non configurables, sont les suivants:

- Visualisation en lecture seule de toutes les cartes définies
- Visualisation en lecture seule des rotations définies dans NagVis (*voir la page sur les rotations dans la documentation NagVis* http://docs.nagvis.org/1.9/en_US/nagvis_config_format_description.html#rotation)
- Modification autorisée de la configuration générale de NagVis

Définition des droits selon les groupes d'utilisateurs

Nom du module: CoreAuthorisationModShinkenGroups

Ce module est très similaire au module "*CoreAuthorisationModGroups*" présent dans une installation NagVis classique. Il permet de définir les droits utilisateurs en fonction des groupes d'utilisateur Shinken dans lequel l'utilisateur est présent.

La configuration des droits se fait grâce au fichier perms.db présent par défaut dans "*/opt/nagvis/etc/*".

Le chemin de ce fichier est configurable dans la configuration de NagVis avec le paramètre "*authorisation_group_perms_file*".

Dans l'exemple, les groupes sont répartis comme suivant:

- Les utilisateurs du groupe "*admins*" ont les droits administrateur dans NagVis
- Les utilisateurs du groupe "*it_admins*" ont accès en lecture et écriture sur toutes les cartes
- Les utilisateurs du groupe "*users*" ont accès en lecture seule à toutes les cartes
- Les utilisateurs du groupe "*users_site1*" ont accès en lecture et écriture seulement sur les cartes "*site1*" et "*site1_bis*"

/opt/nagvis/etc/perms.db

```
{
  "admins": {
    "admin": 1
  },
  "it_admins": {
    "view": [ "*" ],
    "edit": [ "*" ]
  },
  "users": {
    "view": [ "*" ]
  },
  "users_site1": {
    "view": [ "site1", "site1_bis" ],
    "edit": [ "site1", "site1_bis" ]
  }
}
```

La différence de ce module avec celui livré par défaut dans NagVis (*CoreAuthorisationModGroups)* se situe sur le paramètre "admin".

Dans ce module, "admin: 1" aura pour effet de donner tous les droits à l'utilisateur, sauf les droits des gestions des utilisateurs, puisque ceux-ci sont gérés dans Shinken et non dans NagVis.

Authentification avec le widget Page Web

Pour fonctionner avec le widget Page web, Nagvis va nécessiter un paramétrage spécifique (*voir la page* Widget Page web).

- L'authentification se fait avec des cookies.
- Dans les dernières versions de Chrome/Edge et Firefox, l'utilisation de requête "cross-site" par un cookie n'est plus autorisé.
- Cela signifie que si l'application n'est pas installée sur le même serveur que la WebUI de Shinken, l'authentification depuis le widget Page Web ne fonctionnera pas.

Pour palier à ce problème, on peut utiliser HAProxy pour récupérer le flux de ce site et simuler sa présence sur le serveur où est installé la WebUI.

Installation de HAProxy

Pour afficher les cartes de Nagvis qui n'est pas installé sur le serveur où se trouve la WebUI Shinken. Il faut installer HAProxy sur chacun des serveurs où ces cartes doivent être affiché dans le widget Page Web.

RHEL / CentOS 7 - RHEL / Alma / Rocky 8 - RHEL / Alma / Rocky 9

```
yum install haproxy
```

Debian 13

```
apt install haproxy
```

Configuration de SELinux

Pour les systèmes utilisant SELinux (par défaut sous RedHat et ses dérivés)

Ajouter une exception dans SELinux pour HAProxy:

```
setsebool -P haproxy_connect_any=1
```

Configuration de HAProxy

Modifier le fichier de configuration */etc/haproxy/haproxy.cfg* et ajouter à la fin du fichier la configuration de la redirection vers Nagvis :

```
listen  grafana
  bind    SERVEUR_WEBUI:3000
  server    SERVEUR_NAGVIS SERVEUR_NAGVIS:3000
```

Il faut remplacer **SERVEUR_WEBUI** par **0.0.0.0** ou alors l'adresse IP de la machine et **SERVEUR_NAGVIS** par le nom ou l'adresse IP du serveur

Démarrer HAProxy

```
systemctl enable haproxy
systemctl restart haproxy
```

On peut ajouter l'URL suivante dans le widget Page Web : http://SERVEUR_WEBUI:3000/XXXXXXX mais pas ~~http://SERVEUR_NAGVIS:3000/XXXXXXX.~~ La première authentification est nécessaire.

Log HAProxy

Par défaut HAProxy n'a pas de fichier de log. Pour en générer un il faut :

RHEL / CentOS 7 - RHEL / Alma / Rocky 8 - RHEL / Alma / Rocky 9

- Éditer le fichier */etc/rsyslog.conf* et dé-commenter les lignes suivantes :

  ```
  $ModLoad imudp
  $UDPServerRun 514
  ```

- Créer et ajouter dans le fichier */etc/rsyslog.d/haproxy.conf :*

  ```
  local2.* /var/log/haproxy.log
  ```

- Redémarrer rsyslog et haproxy :

  ```
  systemctl restart rsyslog
  systemctl restart haproxy
  ```

- Le fichier de log devrait être en place :

  ```
  tail /var/log/haproxy.log
  ```

Debian 13

- Installer le paquet rsyslog

  ```
  apt install -y rsyslog
  ```

- Éditer le fichier */etc/rsyslog.conf* et dé-commenter les lignes suivantes :

  ```
  module(load="imudp")
  input(type="imudp" port="514")
  ```

- HAProxy fournit par défaut un fichier de configuration pour rsyslog dans */etc/rsyslog.d*. Vérifier qu'il est bel et bien présent :

```
ls -l /etc/rsyslog.d/
```

  - Si ce n'est pas le cas, créer le fichier */etc/rsyslog.d/49-haproxy.conf* et ajouter ce contenu :

```
# Create an additional socket in haproxy's chroot in order to allow logging via
# /dev/log to chroot'ed HAProxy processes
$AddUnixListenSocket /var/lib/haproxy/dev/log

# Send HAProxy messages to a dedicated logfile
:programname, startswith, "haproxy" {
  /var/log/haproxy.log
  stop
}
```

- Redémarrer rsyslog et haproxy :

```
systemctl restart rsyslog
systemctl restart haproxy
```

- Le fichier de log devrait ête en place

```
tail /var/log/haproxy.log
```