Configurez IBNS 2.0 pour des scénarios de seul hôte et de Multi-domaine

Contenu

Introduction

Ce document décrit comment configurer les services 2.0 (IBNS) de réseau basés par identité pour des scénarios de seul hôte et de multi-domaine.

Conditions préalables

Conditions requises

Cisco vous recommande de prendre connaissance des rubriques suivantes :

- Extensible Authentication Protocol au-dessus de réseau local (EAPoL)
- Protocole RADIUS
- Version 2.0 de Logiciel Cisco Identity Services Engine

Composants utilisés

Les informations contenues dans ce document sont basées sur les versions de matériel et de logiciel suivantes :

- Correctif 2 de version 2.0 d'engine de gestion d'identité de Cisco
- Point final avec du SYSTÈME D'EXPLOITATION de Windows 7
- Cisco commutent 3750X avec IOS 15.2(4)E1
- Cisco commutent 3850 avec 03.02.03.SE

- Téléphone IP 9971 de Cisco

Les informations dans ce document sont créées des périphériques dans un environnement de travaux pratiques spécifique. Tous les périphériques utilisés dans ce document ont démarré avec une configuration effacée (par défaut). Si votre réseau est opérationnel, assurez-vous que vous comprenez l'effet potentiel de toute commande.

Configurez

Théorie de configuration

Afin d'activer IBNS 2.0, vous devez exécuter la commande dans le mode privilège sur votre commutateur de Cisco :

```
#authentication display new-style
```

Configurez le switchport pour IBNS 2.0 avec des commandes comme affichées :

```
 access-session host-mode {single-host | multi-domain | multi-auth}
 access-session port-control auto
 dot1x pae authenticator
{mab} service-policy type control subscriber TEST
```

Ces commandes activent l'authentification de dot1x et sur option la dérivation d'authentification MAC (MAB) sur l'interface. Quand vous suivez la nouvelle syntaxe, vous utilisez des commandes qui commence par l'Access-session. Le but de ces commandes correspond pour les commandes qui utilisent l'ancienne syntaxe (commençant par le mot clé d'**authentification**). Appliquez la service-**stratégie** pour spécifier le **policy-map** qui devrait être utilisé pour l'interface.

Le policy-map mentionné ci-dessus définit le comportement du commutateur (authentificateur) pendant l'authentification. Par exemple, vous pouvez spécifier ce qui devrait se produire en cas d'échec d'authentification. Pour chaque **événement** vous pouvez configurer de plusieurs actions basées sur le type de l'événement apparié dans le **class-map** configuré sous lui. Comme exemple, prenez à un regarder la liste comme affiché (**policy-map TEST4**). Si le point final de dot1x qui est connecté à l'interface où cette stratégie est appliquée échoue, alors l'action définie dans **DOT1X_FAILED** est exécutée. Si vous voudriez spécifier le même comportement pour des classes comme **MAB_FAILED** et **DOT1X_FAILED**, alors vous pouvez utiliser la classe par défaut - **class-map toujours.**

```
policy-map type control subscriber TEST4
(...)
 event authentication-failure match-first
  10 class DOT1X_FAILED do-until-failure
   10 terminate dot1x
(...)
  40 class always do-until-failure
   10 terminate mab
   20 terminate dot1x
   30 authentication-restart 60
(...)
```

Le policy-map utilisé pour IBNS 2.0 toujours doit avoir l'**abonné de contrôle de** type.

Vous pouvez visualiser la liste d'événements disponibles de cette façon :

```
Switch(config-event-control-policymap)#event ?
  aaa-available                aaa-available event
  absolute-timeout             absolute timeout event
```

```
  agent-found                  agent found event
  authentication-failure       authentication failure event
  authentication-success       authentication success event
  authorization-failure        authorization failure event
  inactivity-timeout           inactivity timeout event
  session-started              session started event
  tag-added                    tag to apply event
  tag-removed                  tag to remove event
  template-activated           template activated event
  template-activation-failed   template activation failed event
  template-deactivated         template deactivated event
  template-deactivation-failed template deactivation failed event
  timer-expiry                 timer-expiry event
  violation                    session violation event
```

En configuration d'événement vous avez la possibilité pour définir comment des classes devraient être évaluées :

```
Switch(config-event-control-policymap)#event authentication-failure ?
  match-all    Evaluate all the classes
  match-first  Evaluate the first class
```

Vous pouvez définir l'option semblable pour des **class-map**, bien qu'ici vous spécifiiez comment des actions devraient être exécutées au cas où votre classe serait appariée :

```
Switch(config-class-control-policymap)#10 class always ?
  do-all            Execute all the actions
  do-until-failure  Execute actions until one of them fails
  do-until-success  Execute actions until one of them is successful
```

La dernière partie (facultative) de la configuration dans le nouveau style du dot1x est **class-map**. Il devrait également taper l'**abonné de contrôle** et il est utilisé pour apparier le comportement spécifique ou pour trafiquer. Configurez les conditions requises pour l'évaluation d'état de **class-map**. Vous pouvez spécifier que toutes les conditions doivent être appariées ou n'importe quelle condition doit être appariée ou aucune des conditions ne devrait s'assortir.

```
Switch(config)#class-map type control subscriber ?
  match-all   TRUE if everything matches in the class-map
  match-any   TRUE if anything matches in the class-map
  match-none  TRUE if nothing matches in the class-map
```

C'est exemple de **class-map** utilisé pour apparier l'échec d'authentification de dot1x :

```
class-map type control subscriber match-all DOT1X_FAILED
 match method dot1x
 match result-type method dot1x authoritative
```

Pour quelques scénarios, en grande partie quand le service-modèle sont en service, vous devez ajouter la configuration pour la modification de l'autorisation (CoA) :

```
aaa server radius dynamic-author
 client 10.48.17.232 server-key cisco
```

Scénario pour le seul hôte

Diagramme du réseau

Configurations

La configuration de base de 802.1X exigée pour le scénario de seul hôte a testé sur le Catalyst 3750X avec IOS 15.2(4)E1. Le scénario a testé avec le suppliant et le Cisco AnyConnect indigènes de Windows.

```
aaa new-model
!
aaa group server radius tests
 server name RAD-1
!
aaa authentication dot1x default group tests
aaa authorization network default group tests
!
dot1x system-auth-control
!
policy-map type control subscriber TEST
 event session-started match-all
  10 class always do-until-failure
   10 authenticate using dot1x priority 10
!
interface GigabitEthernet1/0/21
 switchport access vlan 613
 switchport mode access
 access-session host-mode single-host
 access-session port-control auto
 dot1x pae authenticator
 service-policy type control subscriber TEST
!
radius server RAD-1
 address ipv4 10.48.17.232 auth-port 1812 acct-port 1813
 key cisco
```

Scénario pour le Multi-domaine

[Diagramme du réseau]

Configurations

le scénario de Multi-domaine a été testé sur le Catalyst 3850 avec IOS 03.02.03.SE dû aux conditions requises PoE (alimentation au-dessus des Ethernets) pour le téléphone IP (téléphone IP 9971 de Cisoc).

```
aaa new-model
!
aaa group server radius tests
 server name RAD-1
!
aaa authentication dot1x default group tests
aaa authorization network default group tests
!
aaa server radius dynamic-author
```

```
 client 10.48.17.232 server-key cisco
!
dot1x system-auth-control
!
class-map type control subscriber match-all DOT1X
 match method dot1x
!
class-map type control subscriber match-all DOT1X_FAILED
 match method dot1x
 match result-type method dot1x authoritative
!
class-map type control subscriber match-all DOT1X_NO_RESP
 match method dot1x
 match result-type method dot1x agent-not-found
!
class-map type control subscriber match-all MAB
 match method mab
!
class-map type control subscriber match-all MAB_FAILED
 match method mab
 match result-type method mab authoritative
!
policy-map type control subscriber TEST4
 event session-started match-all
  10 class always do-until-failure
   10 authenticate using dot1x priority 10
   20 authenticate using mab priority 20
 event authentication-failure match-first
  10 class DOT1X_FAILED do-until-failure
   10 terminate dot1x
  20 class MAB_FAILED do-until-failure
   10 terminate mab
   20 authenticate using dot1x priority 10
  30 class DOT1X_NO_RESP do-until-failure
   10 terminate dot1x
   20 authentication-restart 60
  40 class always do-until-failure
   10 terminate mab
   20 terminate dot1x
   30 authentication-restart 60
 event agent-found match-all
  10 class always do-until-failure
   10 terminate mab
   20 authenticate using dot1x priority 10
 event authentication-success match-all
  10 class always do-until-failure
   10 activate service-template DEFAULT_LINKSEC_POLICY_SHOULD_SECURE
!
interface GigabitEthernet1/0/1
 switchport access vlan 613
 switchport mode access
 switchport voice vlan 612
 access-session host-mode multi-domain
 access-session port-control auto
 mab
 dot1x pae authenticator
 spanning-tree portfast
 service-policy type control subscriber TEST4
!
radius-server attribute 6 on-for-login-auth
radius-server attribute 8 include-in-access-req
radius-server attribute 25 access-request include
radius-server vsa send cisco-nas-port
!
```

```
radius server RAD-1
 address ipv4 10.48.17.232 auth-port 1812 acct-port 1813
 key cisco
```

Vérifiez

Utilisez cette section pour confirmer que votre configuration fonctionne correctement.

Pour la vérification, utilisez ces derniers commandent de répertorier des sessions de tous les switchports :

```
show access-session
```

Vous pouvez également visualiser les informations détaillées au sujet des sessions d'un switchport simple :

```
show access-session interface [Gi 1/0/1] {detail}
```

Dépannez

Cette section fournit des informations que vous pouvez utiliser pour dépanner votre configuration.

Afin de dépanner des questions connexes de 802.1X, vous pouvez activer met au point la même manière que pour la syntaxe de 802.1X de style ancien :

```
debug mab all
debug dot1x all
debug pre all*
```

* optionaly pour mettez- au pointpré vous peut utiliser seulement l'événement et/ou ordonner pour limiter la sortie aux informations pertinentes IBNS 2.0.