Exemple de configuration de fonctions évoluées DAP

Contenu

Introduction

Ce document décrit les fonctions évoluées de Dynamic Access Policies (DAP) pour l'Accès à distance VPN. Vous pouvez utiliser ces fonctions évoluées quand vous avez besoin de flexibilité

supplémentaire d'apparier par des critères.

> Remarque: Référez-vous aux informations importantes sur les commandes de débogage avant d'utiliser les commandes de **débogage**.

Conditions préalables

Conditions requises

Cisco vous recommande de prendre connaissance des rubriques suivantes :

- Une compréhension de DAP de base est exigée. Référez-vous au guide de guide de déploiement ASA 8.x Dynamic Access Policies (DAP) (documentation de support) ou de déploiement ASA 8.x Dynamic Access Policies (DAP) (la communauté de support).
- Une bonne compréhension de la programmation de Lua est également salutaire. Référez-vous aux matériaux de programmation de Lua disponibles sur le Web.

Composants utilisés

Ce document n'est pas limité au logiciel et aux versions de matériel spécifiques, mais Adaptive Security Device Manager (ASDM) est exigé afin de se terminer la configuration.

Les informations contenues dans ce document ont été créées à partir des périphériques d'un environnement de laboratoire 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.

Informations générales

> **Attention** : Utilisez les fonctions faites sur commande de Lua avancées par DAP seulement si la configuration GUI ASDM ou la fonction d'ÉVAL ne fournit pas le comportement assorti que vous avez besoin. Dans des déploiements de production, employez les fonctions avancées de Lua avec soin extrême et avec des conseils de la construction de Cisco/centre d'assistance technique (TAC) afin d'éviter n'importe quel comportement fortuit avec DAP.

Si vous utilisez DAP pour l'Accès à distance VPN, vous pouvez avoir besoin de la flexibilité supplémentaire d'apparier par des critères. Par exemple, vous pouvez appliquer un DAP différent basé sur ces scénarios :

- Une unité organisationnelle (OU) ou tout autre niveau de la hiérarchie contient l'objet utilisateur.
- Un nom de groupe (memberOf) adhère à nommer la convention mais a beaucoup de correspondances possibles, ainsi vous voulez utiliser un masque sur des noms de groupe.
- Vous voulez vérifier l'antivirus, l'anti-spyware, ou les modules de Pare-feu sur le PC de point

final.

1. Employez l'ASDM afin de créer une expression logique pour votre critère de correspondance.

2. Employez le mode avancé afin de créer des fonctions personnalisées avec une expression logique et un code de Lua.

Exemple basé sur OU de correspondance

Le serveur de Protocole LDAP (Lightweight Directory Access Protocol) peut renvoyer beaucoup d'attributs que DAP peut l'utiliser dans une expression logique.

Pour un exemple de ces attributs, utilisez la **commande trace de dap de débogage** sur la console de l'appliance de sécurité adaptable (ASA).

```
assert(function()
if ( (type(aaa.ldap.distinguishedName) == "string") and
(string.find(aaa.ldap.distinguishedName, "OU=Admins,dc=cisco,dc=com$")
~= nil) ) then
return true
end
return false
end
```

Un nom unique (DN) de l'utilisateur est un attribut retourné par le serveur LDAP. Le DN identifie implicitement où l'objet utilisateur se trouve dans le répertoire. Par exemple, si le DN est utilisateur de CN=Joe, OU=Admins, dc=cisco, dc=com, cet utilisateur se trouve dans OU=Admins, dc=cisco, dc=com. Si tous les administrateurs sont dans cette OU (ou tout conteneur au-dessous de ce niveau), employez cette expression logique afin d'apparier sur les critères :

```
assert(function()
if ( (type(aaa.ldap.distinguishedName) == "string") and
(string.find(aaa.ldap.distinguishedName, "OU=Admins,dc=cisco,dc=com$")
~= nil) ) then
return true
end
return false
end)()
```

Dans cet exemple, la fonction string.find tient compte d'une expression régulière. Le $ à l'extrémité de la chaîne ancre cette chaîne à l'extrémité du champ de distinguishedName.

Employez le caractère d'échappement % dans votre chaîne de recherche afin d'échapper à des caractères particuliers comme (). % + - * ? [^ $. Par exemple, vous pouvez échapper - au caractère dans cette chaîne (OU=Admins, dc=my-domain, dc=com$) suivant les indications de cette chaîne (OU=Admins, dc=my%-domain, dc=com$).

Exemple d'adhésion à des associations (memberOf)

Vous pouvez créer une expression logique semblable et de base pour le filtrage de l'adhésion à

des associations de Répertoire actif (AD). Puisque les utilisateurs peuvent être des membres de plusieurs groupes, DAP analyse la réponse du serveur LDAP dans les entrées distinctes et les tient dans une table. Dans ce cas, une fonction plus évoluée est exigée :

- Comparez le champ de memberOf comme chaîne si l'utilisateur appartient à seulement un groupe.
- Réitérez par chaque champ retourné de memberOf si les données renvoyées sont de type « table. »

Par exemple, si un utilisateur est un membre de n'importe quel groupe avec lequel finit « - stu, » ils apparient ce DAP :

```
assert(function()
local pattern = "-stu$"
local attribute = aaa.ldap.memberOf
if ((type(attribute) == "string") and
(string.find(attribute, pattern) ~= nil)) then
return true
elseif (type(attribute) == "table") then
local k, v
for k, v in pairs(attribute) do
if (string.find(v, pattern) ~= nil) then
return true
end
end
end
return false
end)()
```

CheckAndMsg avec la fonction personnalisée

Cette fonction emploie DAP avec le positionnement d'action pour se terminer :

```
(assert(function()
local block_connection = true
local update_threshold = "150000" --this is the value of lastupdate in
seconds
for k,v in pairs(endpoint.av) do
if (CheckAndMsg(EVAL(v.exists, "EQ", "true", "string") and EVAL
(v.lastupdate, "LT", update_threshhold, "integer"),k.." exists; last update is
"..string.sub((tonumber(v.lastupdate)/86400), 1, 3).." days",k.." does not exist; last update is
"..string.sub((tonumber(v.lastupdate)/86400), 1, 3).." days")) then
block_connection = false
end
end
return block_connection
end)())
```

Sur l'arrêt, il affiche ce message :

```
Login denied.<AV Name> does not exists; last update is <X> days
```

Antivirus, anti-spyware, et exemples de Pare-feu

Ces fonctions de Lua vérifient des attributs liés à l'antivirus, anti-spyware, et les modules de Pare-feu sur le PC de point final retourné par l'hôte du Cisco Secure Desktop (CSD) balayent.

Vérifiez l'installation d'antivirus

Cette fonction personnalisée vérifie si le CSD détecte n'importe quel antivirus :

```
assert(function()
for k,v in pairs(endpoint.av) do
if (EVAL(v.exists, "EQ", "true", "string")) then
return true
end
end
return false
end)()
```

Vérifiez l'installation et la dernière modification d'antivirus, et fournissez les messages d'erreur

Cet exemple explique comment DAP peut vérifier une installation d'antivirus, vérifier la dernière modification, et informer l'utilisateur pour la correction. Il utilise une fonction semblable à celle vérifient en test l'installation d'antivirus.

Placez l'Authentification, autorisation et comptabilité (AAA) vous attribue souhait pour apparier. Dans le domaine avancé, assurez qu'ET l'exécution est sélectionnée ; dans le champ action, assurez que l'option de terminaison est sélectionnée. Si l'utilisateur apparie les attributs d'AAA et si la fonction de Lua renvoie une valeur de vrai, DAP est sélectionné, un message apparaît qui explique pourquoi l'enregistrement DAP a été affiché, et la connexion utilisateur est terminée. Si la fonction de Lua ne renvoie pas une valeur de vrai, DAP ne s'assortit pas et accès d'autorisations. Dans le domaine de zone de message, entrez le message, « aucun programme d'antivirus trouvé, installez s'il vous plaît l'antivirus et l'essai de nouveau. » Si l'utilisateur a un module d'antivirus et est au-dessous du seuil de jours de mise à jour, ils ne sont pas donnés un message comme indiqué par les guillemets dans la ligne 7 de cet exemple :

```
(assert(function()
local block_connection = true
local update_days = "15" --days
local av_lastupdate = update_days*86400
for k,v in pairs(endpoint.av) do
if (CheckAndMsg(EVAL(v.exists, "EQ", "true", "string") and EVAL(v.lastupdate, "LT",
av_lastupdate, "integer"),"",k.." exists; but last update is greater than 15 days old. Expecting
under 15 days.")) then
block_connection = false
elseif (EVAL(v.exists, "NE", "true", "string")) then
block_connection = true
end
end
return block_connection
end)())
```

Si l'utilisateur a l'antivirus de Norton, mais la dernière modification est plus grande que 15 jours, ce message témoin apparaît :

```
NortonAV exists; but last update is greater than 15 days old. Expecting under 15 days.
```

Si l'ÉVAL ne s'assortit pas, il va à la prochaine fonction, des correspondances, et renvoie une valeur de vrai. Puisqu'il n'y a aucun CheckAndMsg associé avec la deuxième fonction, elle utilise le texte du message DAP :

```
No anti-virus program found, please install anti-virus and try again.
```

En résumé, DAP recherche un AAA d'utilisateur et un attribut de point final afin d'apparier DAP. Si

DAP s'assortit, l'utilisateur est terminé avec un message. La correspondance de point final est un résultat d'un ÉVAL de Lua qui renvoie vrai ou faux à DAP. Les correspondances vraies et refuse la connexion. Un faux n'apparie pas et permet la connexion.

1. La première fonction dans les contrôles par écho si endpoint.av.xxxxx.exists est égal pour rectifier et si la dernière modification est moins que les jours configurés. On permet à des utilisateurs sans logiciel antivirus l'accès, parce que les correspondances d'AAA d'utilisateur, mais le Lua regarde spécifiquement pour endpoint.av.xxxxx.exists = rectifient et jours de <= endpoint.av.xxxxx.lastupdate.
2. La deuxième boucle attrape des utilisateurs sans logiciel antivirus et les bloque, parce que la deuxième fonction regarde seulement pour le Ne endpoint.av.xxxxx.exists vrai. Si l'utilisateur que le point final poids du commerce existe n'est pas égal pour rectifier, la fonction renvoie une valeur de vrai, qui signifie qu'elles n'ont pas l'antivirus. DAP apparie et refuse la connexion.

Vérifiez l'installation d'anti-spyware

Cette fonction personnalisée vérifie si le CSD détecte l'anti-spyware :

```
assert(function()
for k,v in pairs(endpoint.as) do
if (EVAL(v.exists, "EQ", "true", "string")) then
return true
end
end
return false
end)()
```

Vérifiez l'installation de Pare-feu

Cette fonction personnalisée vérifie si le CSD détecte un Pare-feu :

```
assert(function()
for k,v in pairs(endpoint.fw) do
if (EVAL(v.exists, "EQ", "true", "string")) then
return true
end
end
return false
end)()
```

Vérifiez l'antivirus, l'anti-spyware, ou l'installation de Pare-feu

Cette fonction renvoie un vrai si un antivirus, l'anti-spyware ou un module de Pare-feu est trouvé :

```
assert(function()
function check(antix)
if (type(antix) == "table") then
for k,v in pairs(antix) do
if (EVAL(v.exists, "EQ", "true", "string")) then
return true
end
end
end
return false
```

```
end
return (check(endpoint.av) or check(endpoint.fw) or check(endpoint.as))
end)()
```

Terminez si aucune installation de logiciel espion

La seule différence entre cette fonction et la fonction vérifient en test l'installation d'anti-spyware est que « pas » précède l'affirmation.

```
not assert(function()
for k,v in pairs(endpoint.as) do
if (EVAL(v.exists, "EQ", "true", "string")) then
return true
end
end
return false
end)()
```

Vérifiez l'antivirus et l'installation de Pare-feu, et validez le dernier antivirus que la mise à jour n'est pas plus grande que trente jours

Cet exemple renvoie vrai si un antivirus et un Pare-feu sont trouvés et si la dernière modification de l'antivirus n'est pas plus grande que 30 jours :

```
assert(function()
function checkav(antix)
if (type(antix) == "table") then
for k,v in pairs(antix) do
if (EVAL(v.activescan, "EQ", "ok", "string") and EVAL (v.lastupdate, "LT", "2592000",
"integer")) then
return true
end
end
end
return false
end
function checkfw(antix)
if (type(antix) == "table") then
for k,v in pairs(antix) do
if (EVAL(v.enabled, "EQ", "ok", "string")) then
return true
end
end
end
return false
end
return (checkav(endpoint.av) and checkfw(endpoint.fw))
end)()
```

Puisque le Pare-feu n'a pas une valeur de lastupdate à retourner, il a une fonction distincte.

Apparier d'expression régulière

Cette section décrit les fonctions qui emploient des expressions d'expression régulière afin d'apparier certains attributs et déterminer la validité de l'ordinateur hôte. Ces capacités d'expression régulière ont été testées et sont valides :

- Le symbole dollar ($) ancre la chaîne de recherche à la fin de la valeur retournée.
- Le caret (^) ancre la chaîne de recherche au début de la valeur retournée.
- Caractères encadrés, comme [aa], plusieurs caractères de correspondance en position spécifique. Par exemple, afin d'apparier OU=Cisco (ne distinguant pas majuscules et minuscules), utilisation OU= [cc] [II] [solides solubles] [cc] [Oo].
- La période (.) apparie n'importe quel caractère unique en cette position. Par exemple, groupe. Correspondances Group01Users d'utilisateurs, Group33Users, et ainsi de suite.

Connectez si le PC de point final a n'importe quel exemple du correctif KB944

Cette fonction utilise l'expression régulière s'assortissant afin de voir si la liste de correctif contient un modèle. Dans cet exemple, le Cisco Secure Desktop renvoie tous les correctifs sur le PC de point final ; s'il y a un exemple de KB944, la stratégie DAP s'assortit et est imposée.

```
assert(function ()
local pattern = "KB944"
local true_on_match = true
local match = false
for k,v in pairs(endpoint.os.hotfix) do
print(k)
match = string.find(k, pattern)
if (match) then
if (true_on_match) then
return true
else return (false)
end
end
end
end)()
```

Par exemple, si l'ordinateur hôte a le correctif KB944533 ou le correctif KB944653, il apparie la règle.

Script de l'utilisation LUA pour vérifier OUI de l'adresse MAC

Cette fonction est semblable à celle décrite dedans se connectent si le PC de point final a n'importe quel exemple du correctif KB944. Cette fonction emploie une expression régulière afin d'apparier administrativement l'identifiant unique (OUI) de l'adresse MAC.

Dans cet exemple, les débuts d'adresse MAC avec d067.e5XX.XX. Employez une expression régulière et un code de Lua afin d'apparier les ordinateurs qui démarrent avec le même MAC OUI.

```
assert(function ()
local pattern = "^d067\.e5*"
local true_on_match = true

local match = false
for k,v in pairs(endpoint.device.MAC) do
print(k)
match = string.find(k, pattern)
if (match) then
if (true_on_match) then
return true
else return (false)
end
end
```

```
end
end)()
```

Remarque: Une différente version de cette fonction est exigée pour vérifier à valeurs multiples.

Connectez basé sur les trois premières lettres de l'adresse Internet (non distinguant majuscules et minuscules)

Cette fonction emploie des expressions régulières afin de déterminer si les trois premières lettres de l'adresse Internet sont mSv (ne distinguant pas majuscules et minuscules) :

```
assert(function()
local match_pattern = "^[Mm][Ss][Vv]"
local match_value = endpoint.device.hostname
if (type(match_value) == "string") then
if (string.find(match_value, match_pattern) ~= nil) then
return true
end
elseif (type(match_value) == "table") then
local k,v
for k,v in pairs(match_value) do
if (string.find(v, match_pattern) ~= nil) then
return true
end
end
end
return false
end)()
```

Connectez si Device.id de PC de point final et de numéro de série sur le certificat sont identiques

Cette expression de Lua est censée pour se connecter si les device.id du PC de point final et du numéro de série sur le certificat sont identiques :

```
assert(function()
local match_pattern = endpoint.device.id
for k,v in pairs(endpoint.certificate.user) do
if (type(v.subject_e) == "string") then
if (string.find(v.subject_e, match_pattern) ~= nil) then
return true
end
elseif (type(v.subject_e) == "table") then
local k,v
for k,v in pairs(v.subject_e) do
if (string.find(v, match_pattern) ~= nil) then
return true
end
end
end
return false
end ) ()
```

Remarque: L'utilisation du caractre générique (*****) ne fonctionne pas dans cette fonction particulière (endpoint.certificate.user [« * »] ne fait pas travail). Vous devez récupérer chaque paire kilovolt individuellement et l'analyser par elles.

Imposez DAP basé sur le balayage d'hôte CSD pour la clé d'enregistrement du domaine

Cette procédure fournit à un exemple d'une procédure de configuration l'ASDM.

1. Localisez la clé de registre qui tient à de domaine \ HKEY_LOCAL_MACHINE \ SYSTÈME \ CurrentControlSet \ services \ Tcpip \ paramètres \ domaine.

2. Définissez le paramètre de balayage d'hôte pour les paramètres de registre.

3. Appliquez-vous l'attribut de point final de registre à la stratégie DAP.

4. Établissez la session VPN de Secure Sockets Layer (SSL).

5. Vérifiez l'application de stratégie DAP par l'intermédiaire de DAP met au point.

Soutien DAP du Windows 7 et du CSD 3.5

Des Plateformes de Windows 7 sont prises en charge avec version 3.5 CSD ou plus tard. Avec la release de maintenance ASDM 6.2.x et les releases 6.3.x, vous pouvez directement employer l'interface afin de vérifier le SYSTÈME D'EXPLOITATION de Windows 7. Avec des releases plus tôt ASDM, un script avancé DAP Lua est exigé afin de vérifier des ordinateurs de Windows 7. Sur une ASA avec la version 8.x et la pré-bêta version 3.5 CSD, écrivez cette chaîne de script de Lua dans la case avancée par DAP ASDM afin d'exécuter vérifie des ordinateurs de Windows 7 :

```
(EVAL(endpoint.os.version,"EQ","Windows 7","string"))
```

Identification des iPhones, des iPads, et des périphériques mobiles

Cette expression de Lua vous permet de dépister les périphériques mobiles spécifiques par leurs identifiants uniques (uid). Vous pouvez employer DAP afin de réaliser cette fonctionnalité de base.

Quand la valeur ne peut pas êtes codé en dur et les besoins d'être lu de l'AD, ceci devient plus difficile. Puisqu'il n'y a aucun champ spécifique d'UID dans l'AD, vous pouvez enregistrer la valeur pour un utilisateur particulier sous un champ différent. Cet exemple emploie l'otherHomePhone pour enregistrer l'UID.

Pour vous aider à identifier l'UID pour un iPhone ou un iPad, recherchez le Web pour un outil approprié.

Une fois que vous identifiez l'UID, ajoutez-le à l'otherHomePhone dans l'entrée d'AD pour cet utilisateur.

De la commande du **LDAP 255 de débogage** et du test d'authentification d'utilisateur, obtenez l'attribut de LDAP étant poussé, qui est otherHomePhone.

Permettez au téléphone pour se connecter, puis exécutez un suivi DAP pendant la connexion tentée afin d'identifier l'attribut de point final qui contient l'UID (endpoint.anyconnect.deviceuniqueid).

Cette expression de Lua peut alors comparer les deux paramètres :

```
assert(function()
if (type(aaa.ldap.otherHomePhone) ==type(endpoint.anyconnect.deviceuniqueid)
then
return true
end
return false
end)()
```

Utilisation de DAP d'empêcher la connexion par le navigateur spécifique

Cette procédure décrit comment employer DAP pour empêcher la connexion par un navigateur de Chrome :

1. Enable CSD.

2. Sous la configuration de balayage d'hôte, employez l'ID de processus (PID) et le nom du processus afin d'ajouter un balayage de processus.

   Vous pouvez déterminer le PID et le nom du processus sur Windows avec le gestionnaire de tâches. Afin d'afficher la valeur PID, ouvrir le **gestionnaire de tâches**, allez aux **processus l'**onglet, cliquez sur le **menu Affichage**, puis cliquez sur les **colonnes choisies**. Dans les colonnes choisies ou la page de processus choisie les colonnes dialoguent, font tic tac et vérifient la case à cocher pour **PID (identifiant de processus)**, et cliquent sur OK.

   Sur des MACs, vous pouvez déterminer l'ID de processus avec le moniteur d'activité. Ou, dans le shell de coup (que vous pouvez mettre en boîte également utilisez dans Unix), employez la **picoseconde -** commande **e** tandis que le processus s'exécute, puis pour apparier les PID au nom du processus avec la commande de **/proc/ <PID>/cmdline de cat**.

3. Créez une stratégie DAP afin de tester, notamment, si ce processus s'exécute sur

l'ordinateur.

4. Testez votre connexion.

Mises en garde

1. Le problème avec cette solution est que l'utilisateur ne peut pas avoir Chrome ouvert sur son ordinateur du tout. DAP vérifie simplement si ce processus de Chrome de détail s'exécute, mais ne vérifie pas pour voir si la session sans client était initiée par ce processus ou par un autre processus. Ainsi l'utilisateur ne peut pas exécuter Chrome à l'arrière-plan quand une connexion de webvpn est tentée.
2. Assumez un scénario où l'utilisateur emploie Firefox pour ouvrir une session de webvpn, et la tentative de procédure de connexion échoue. L'utilisateur se souvient que Chrome s'exécute toujours, ainsi l'utilisateur termine la connexion et les essais de Chrome pour ouvrir une session de nouveau. La procédure de connexion échoue toujours parce que le CSD doit réexécuter le balayage d'hôte. Ainsi, l'utilisateur doit également clôturer l'exemple de Firefox qui a été utilisé pour accéder au webvpn, puis redémarrent Firefox. Ce processus peut être embrouillant à l'utilisateur. Cisco recommande que vous créiez un message d'échec DAP qui indique l'utilisateur terminer Chrome et fermer le navigateur qu'ils utilisent actuellement :

FORUM AUX QUESTIONS

Cette section apporte une réponse à une des questions fréquemment posées considère dedans les informations qui sont décrites dans ce document.

Pourquoi mon script LUA fonctionne-t-il pour quelques utilisateurs, mais pas pour tous ?

Examinez ce script LUA :

```
assert(function()
     for k,v in pairs(endpoint.certificate.user) do
         if (v.subject_store == "capi" and v.subject_dc == "homedepot") then
             return true
         end
     end
     return false
 end)()
```

Ce script est conçu afin d'apparier la mémoire de certificat et le C.C de sujet qui est trouvé dans le certificat. Cependant, ce script a été testé avec de plusieurs ordinateurs et avéré pour travailler à quelques ordinateurs, mais pour échouer sur beaucoup d'autres.

La raison pour laquelle ce script fonctionne seulement est par intermittence en raison de la manière dont hostscan renvoie les valeurs. Quand vous visualisez le suivi DAP qui ne fonctionne pas, vous pouvez voir que le *subject_dc* renvoie de plusieurs valeurs par certificat. Vous pouvez également voir que la dernière valeur retournée n'est pas *Home Depot* :

```
DAP_TRACE: endpoint.policy.location = "CORP-Windows"
:
.
DAP_TRACE: endpoint.certificate.user["6"].subject_store = "capi"
DAP_TRACE: endpoint.certificate.user["6"].subject_dc = "com"
DAP_TRACE: endpoint.certificate.user["6"].subject_dc = "homedepot"
DAP_TRACE: endpoint.certificate.user["6"].subject_dc = "amer"
```

Quand vous visualisez le suivi DAP qui fonctionne, ceci peut être observé :

```
DAP_TRACE: endpoint.certificate.user["20"] = {}
DAP_TRACE: endpoint.certificate.user["20"].subject_cn = "JHD0C6"
DAP_TRACE: endpoint.certificate.user["20"].subject_e = "jimmie_harden@homedepot.com"
DAP_TRACE: endpoint.certificate.user["20"].subject_ou = "Associates"
DAP_TRACE: endpoint.certificate.user["20"].subject_store = "capi"
DAP_TRACE: endpoint.certificate.user["20"].subject_dc = "com"
DAP_TRACE: endpoint.certificate.user["20"].subject_dc = "homedepot"
DAP_TRACE: endpoint.certificate.user["20"].issuer_cn = "The Home Depot Remote
Access Issuing CA v2"
```

Ceci indique que le script LUA fonctionne correctement. Cependant, en raison de la manière dont la posture-estimation renvoie les valeurs sur quelques ordinateurs, le script ne s'assortit pas. Dans ce cas, les difficultés pour les id CSCuu85646 de bogue Cisco et CSCuh67472 sont exigés.