Guide de dépannage de l'authentificateur RADIUS non valide et de l'authentificateur de message

Contenu

Introduction

Ce document décrit deux mécanismes de sécurité RADIUS :

- En-tête de l'authentificateur
- Attribut Message-Authenticator

Ce document couvre ce que sont ces mécanismes de sécurité, comment ils sont utilisés et quand vous devez vous attendre à un échec de validation.

En-tête de l'authentificateur

Selon RFC 2865, l'en-tête Authenticator a une longueur de 16 octets. Lorsqu'il est utilisé dans une demande d'accès, il est appelé authentificateur de demande. Lorsqu'il est utilisé dans n'importe quel type de réponse, il est appelé authentificateur de réponse. Il est utilisé pour :

- Authentification de la réponse
- Masquage de mot de passe

Authentification de la réponse

Si le serveur répond avec l'authentificateur de réponse correct, le client peut calculer si cette réponse était liée à une requête valide.

Le client envoie la demande avec l'en-tête d'authentificateur aléatoire. Ensuite, le serveur qui envoie la réponse calcule l'authentificateur de réponse à l'aide du paquet de requête et du secret partagé :

```
ResponseAuth = MD5(Code + ID + Length + RequestAuth + Attributes + Secret)
```

Le client qui reçoit la réponse effectue la même opération. Si le résultat est le même, le paquet est correct.

> **Note**: L'attaquant qui connaît la valeur secrète ne peut pas usurper la réponse à moins qu'il ne soit capable de détecter la requête.

Quand devez-vous attendre un échec de validation ?

L'échec de validation se produit si le commutateur ne met plus en cache la demande (par exemple, en raison d'un délai d'attente). Vous pouvez également l'observer lorsque le secret partagé n'est pas valide (oui - Access-Reject inclut également cet en-tête). De cette manière, le périphérique d'accès au réseau (NAD) peut détecter la non-correspondance de secret partagé. Généralement, il est signalé par les clients/serveurs AAA (Authentication, Authorization, and Accounting) comme une incompatibilité de clé partagée, mais il ne révèle pas les détails.

Masquage du mot de passe

L'en-tête Authenticator est également utilisé afin d'éviter d'envoyer l'attribut User-Password en texte brut. Tout d'abord, le Message Digest 5 (MD5 - secret, authentificateur) est calculé. Ensuite, plusieurs opérations XOR avec les blocs du mot de passe sont exécutées. Cette méthode est sensible aux attaques hors ligne (tables arc-en-ciel) car MD5 n'est plus perçu comme un algorithme unidirectionnel puissant.

Voici le script Python qui calcule le mot de passe utilisateur :

```
def Encrypt_Pass(password, authenticator, secret):
       m = md5()
       m.update(secret+authenticator)
       return "".join(chr(ord(x) ^ ord(y)) for x, y in zip(password.ljust
(16,'\0')[:16], m.digest()[:16]))
```

Retransmissions

Si l'un des attributs de la demande d'accès RADIUS a changé (comme l'ID RADIUS, le nom d'utilisateur, etc.), le nouveau champ Authenticator doit être généré et tous les autres champs qui en dépendent doivent être recalculés. S'il s'agit d'une retransmission, rien ne devrait changer.

Gestion de comptes

La signification de l'en-tête Authenticator est différente pour une demande d'accès et une demande de comptabilité.

Pour une demande d'accès, l'authentificateur est généré aléatoirement et il est attendu qu'il reçoive une réponse avec ResponseAuthenticator calculée correctement, ce qui prouve que la réponse était liée à cette demande spécifique.

Pour une demande de compte, l'authentificateur n'est pas aléatoire, mais il est calculé (conformément à la RFC 2866) :

```
RequestAuth = MD5(Code + ID + Length + 16 zero octets + Attributes + Secret)
```

De cette manière, le serveur peut vérifier immédiatement le message de comptabilité et supprimer le paquet si la valeur recalculée ne correspond pas à la valeur Authenticator. Le moteur ISE (Identity Services Engine) renvoie :

```
11038 RADIUS Accounting-Request header contains invalid Authenticator field
```

La raison typique est la clé secrète partagée incorrecte.

Attribut Message-Authenticator

L'attribut Message-Authenticator est l'attribut RADIUS défini dans le document RFC 3579. Il est utilisé dans un but similaire : pour signer et valider. Mais cette fois, il n'est pas utilisé pour valider une réponse mais une requête.

Le client qui envoie une demande d'accès (il peut également s'agir d'un serveur qui répond par un Access-Challenge) calcule le code HMAC (Hash-Based Message Authentication Code)-MD5 à partir de son propre paquet, puis ajoute l'attribut Message-Authenticator comme signature. Ensuite, le serveur peut vérifier qu'il effectue la même opération.

La formule ressemble à l'en-tête Authenticator :

```
Message-Authenticator = HMAC-MD5 (Type, Identifier, Length, Request Authenticator,
Attributes)
```

La fonction HMAC-MD5 prend en compte deux arguments :

- Charge utile du paquet, qui inclut le champ Message-Authenticator de 16 octets rempli de zéros
- Le secret partagé

Quand l'authentificateur de message doit-il être utilisé ?

L'authentificateur de message DOIT être utilisé pour chaque paquet, qui inclut le message EAP (Extensible Authentication Protocol) (RFC 3579). Cela inclut à la fois le client qui envoie la demande d'accès et le serveur qui répond avec la demande d'accès. L'autre côté doit supprimer silencieusement le paquet si la validation échoue.

Quand devez-vous attendre un échec de validation ?

Échec de validation lorsque le secret partagé n'est pas valide. Ensuite, le serveur AAA ne peut pas valider la demande.

L'ISE rapporte :

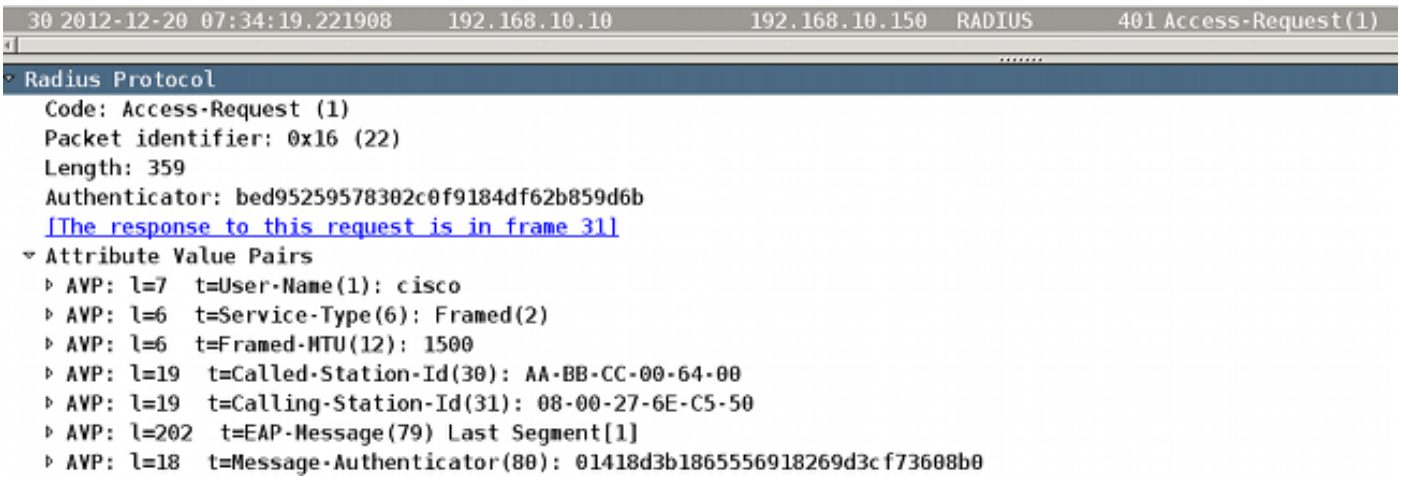
```
11036 The Message-Authenticator Radius Attribute is invalid.
```

Cela se produit généralement à une étape ultérieure lorsque le message EAP est joint. Le premier paquet RADIUS de la session 802.1x n'inclut pas le message EAP ; il n'existe aucun champ Message-Authenticator et il n'est pas possible de vérifier la demande, mais à ce stade, le client est en mesure de valider la réponse à l'aide du champ Authenticator.

Valider l'attribut Message-Authenticator

Voici un exemple pour illustrer comment vous comptez manuellement la valeur afin de vous assurer qu'elle est calculée correctement.

Le paquet numéro 30 (Access-Request) a été choisi. Il se trouve au milieu de la session EAP et le paquet inclut le champ Message-Authenticator. L'objectif est de vérifier que l'authentificateur de message est correct :

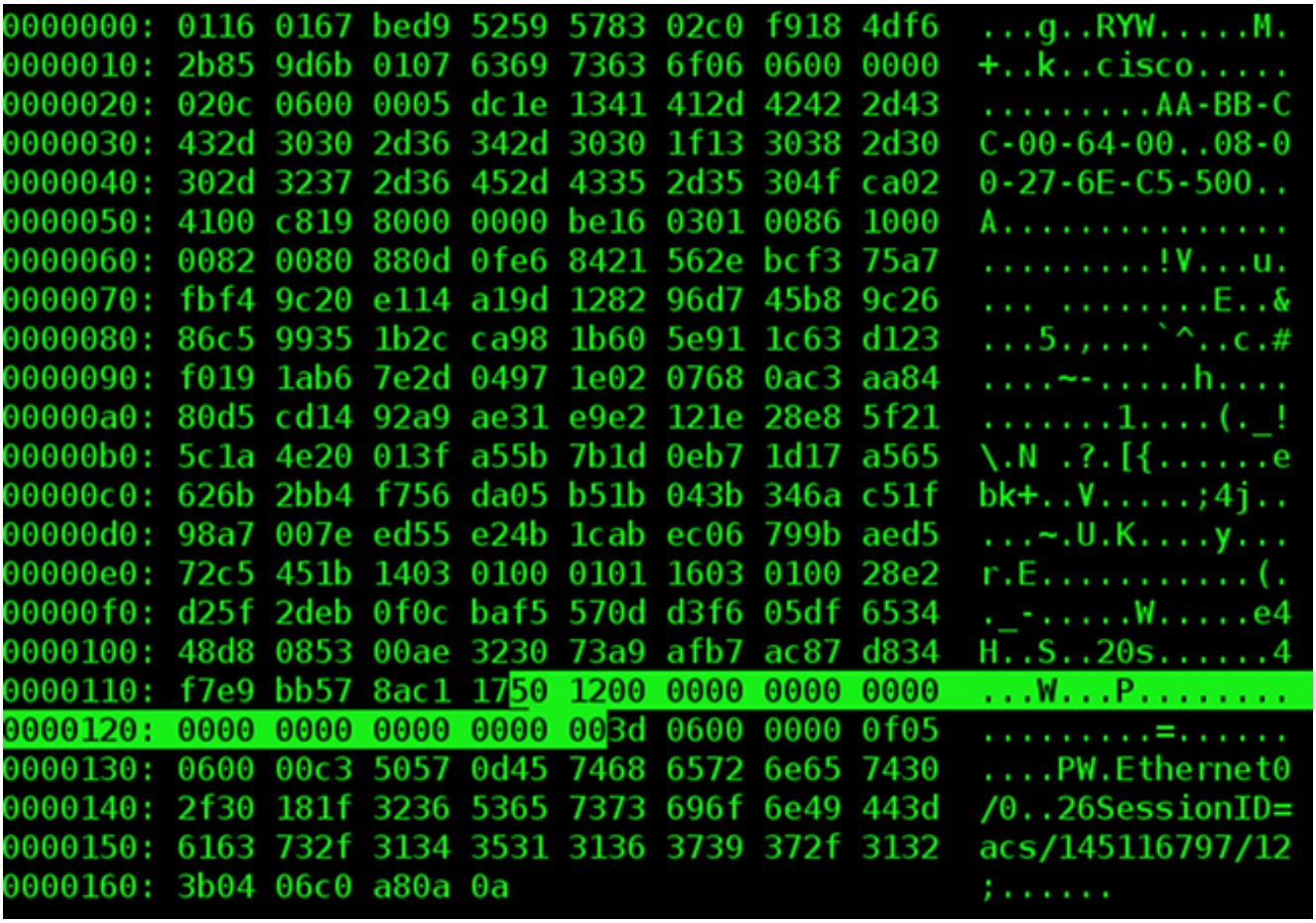


1. Cliquez avec le bouton droit sur **Radius Protocol** et choisissez **Exporter les octets de paquets sélectionnés**.

2. Écrivez cette charge utile RADIUS dans un fichier (données binaires).

3. Pour calculer le champ Message-Authenticator, vous devez y mettre des zéros et calculer le HMAC-MD5.

   Par exemple, lorsque vous utilisez un éditeur hexadécimal/binaire, tel que vim, après avoir tapé " :%!xxd », qui passe en mode hexadécimal et fait zéro 16 octets commençant après « 5012 » (50hex fait 80 dec de type Message-Authenticator, et 12 est la taille qui est 18 incluant l'en-tête AVP) :



Après cette modification, la charge utile est prête. Il est nécessaire de revenir au mode hexadécimal/binaire (type : " :%!xxd -r ») et enregistrez le fichier (« :wq »).

4. Utilisez OpenSSL afin de calculer HMAC-MD5 :

```
pluton # cat packet30-clear-msgauth.bin | openssl dgst -md5 -hmac 'cisco'
(stdin)= 01418d3b1865556918269d3cf73608b0
```

La fonction HMAD-MD5 prend deux arguments : la première entrée standard (stdin) est le message lui-même et la seconde est le secret partagé (Cisco dans cet exemple). Le résultat est exactement la même valeur que le Message-Authenticator attaché au paquet de requête d'accès RADIUS.

Il en va de même avec le script Python :

```
pluton # cat hmac.py
#!/usr/bin/env python

import base64
import hmac
import hashlib

f = open('packet30-clear-msgauth.bin', 'rb')
try:
   body = f.read()
finally:
   f.close()

digest = hmac.new('cisco', body, hashlib.md5)
```

```
d=digest.hexdigest()
print d

pluton # python hmac.py
01418d3b1865556918269d3cf73608b0
```

L'exemple précédent montre comment calculer le champ Message-Authenticator à partir de Access-Request. Pour Access-Challenge, Access-Accept et Access-Reject, la logique est exactement la même, mais il est important de se rappeler que Request Authenticator doit être utilisé, ce qui est fourni dans le paquet Access-Request précédent.

Informations connexes

- RFC 2865
- RFC 2866
- RFC 3579
- Support et documentation techniques - Cisco Systems