RADIUS無效驗證器和消息驗證器故障排除指南

目錄

簡介

本檔案介紹兩種RADIUS安全機制：

- 驗證器標頭
- Message-Authenticator屬性

本文檔介紹這些安全機制的用途、使用方式以及驗證失敗預期發生時間。

驗證器標頭

根據RFC 2865，驗證器標頭長度為16位元組。在訪問請求中使用時，稱為請求驗證器。當它用於任何型別的響應時，它稱為響應驗證器。它用於：

- 響應驗證
- 密碼隱藏

響應驗證

如果伺服器使用正確的響應身份驗證器進行響應，則客戶端可以計算該響應是否與有效請求相關。

客戶端傳送帶有隨機身份驗證器標頭的請求。然後，傳送響應的伺服器將使用請求資料包以及共用金鑰來計算響應身份驗證器：

```
ResponseAuth = MD5(Code + ID + Length + RequestAuth + Attributes + Secret)
```
接收響應的客戶端執行相同的操作。如果結果相同，則表示資料包正確。

附註：知道秘密值的攻擊者無法欺騙響應，除非它能夠嗅探請求。

什麼時候驗證會失敗？

如果交換機不再快取請求（例如，由於超時），則會出現驗證失敗。 當共用金鑰無效時，您也可能遇到此情況（是 — Access-Reject也包含此標頭）。 如此，網路存取裝置(NAD)可以偵測共用密碼不相符。通常，身份驗證、授權和記帳(AAA)客戶端/伺服器會將其報告為共用金鑰不匹配，但不會顯示詳細資訊。

密碼隱藏

身份驗證器標頭也用於避免以純文字檔案格式傳送使用者密碼屬性。首先計算消息摘要5(MD5 - secret， authenticator)。然後執行多個帶有密碼塊的XOR操作。此方法容易受到離線攻擊（彩虹表），因為MD5不再被視為一個強大的單向演算法。

以下是用於計算使用者密碼的Python指令碼：

```
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]))
```

重新傳輸

如果RADIUS Access-Request中的任何屬性已更改（例如RADIUS ID、使用者名稱等），應生成新的Authenticator欄位，並應重新計算依賴該欄位的所有其他欄位。如果是重傳，則無需更改。

會計

Access-Request和Accounting-Request的Authenticator Header的含義不同。

對於Access-Request，Authenticator是隨機生成的，它需要接收響應，並且ResponseAuthenticator計算正確，這證明了響應與該特定請求相關。

對於Accounting-Request，Authenticator不是隨機的，但它是計算的（根據RFC 2866）：

```
RequestAuth = MD5(Code + ID + Length + 16 zero octets + Attributes + Secret)
```
這樣，如果重新計算的值與Authenticator值不匹配，伺服器可以立即檢查記帳消息並丟棄資料包。身份服務引擎(ISE)返回：

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

此問題的典型原因是共用金鑰不正確。

消息驗證器屬性

Message-Authenticator屬性是在RFC 3579中定義的RADIUS屬性。其用途類似：進行簽名和驗證。但這一次，它不是用於驗證響應，而是用於驗證請求。

傳送存取要求的使用者端（也可以是回應Access-Challenge的伺服器）會根據自己的封包計算基於雜湊的訊息驗證碼(HMAC)-MD5，然後新增訊息驗證器屬性作為簽名。然後，伺服器可以驗證它是否執行了相同的操作。

此公式類似於身份驗證器標頭：

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

HMAC-MD5函式有兩個引數：

- 資料包的負載，包括用零填充的16位元組消息驗證器欄位
- 共用金鑰

何時應使用Message-Authenticator?

每個資料包都必須使用Message-Authenticator，其中包括可擴展身份驗證協定(EAP)消息(RFC 3579)。 這包括傳送訪問請求的客戶端和使用Access-Challenge進行響應的伺服器。如果驗證失敗，另一端應靜默丟棄資料包。

什麼時候驗證會失敗？

當共用金鑰無效時，將發生驗證失敗。然後，AAA伺服器無法驗證請求。

ISE報告：

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

這通常發生在附加EAP消息的較後階段。802.1x作業階段的首個RADIUS封包不包括EAP訊息；沒有Message-Authenticator欄位，並且無法驗證請求，但在此階段，客戶端可以使用Authenticator欄位驗證響應。

驗證Message-Authenticator屬性

以下範例顯示如何手動計數值，以確保正確計算值。

已選擇資料包編號30(Access-Request)。它位於EAP會話的中間，資料包包括Message-Authenticator欄位。目的是驗證Message-Authenticator是否正確：

```
30 2012-12-20 07:34:19.221908    192.168.10.10          192.168.10.150  RADIUS      401 Access-Request(1)
```

```
Radius Protocol
   Code: Access-Request (1)
   Packet identifier: 0x16 (22)
   Length: 359
   Authenticator: bed95259578302c0f9184df62b859d6b
   [The response to this request is in frame 31]
 Attribute Value Pairs
   AVP: l=7  t=User-Name(1): cisco
   AVP: l=6  t=Service-Type(6): Framed(2)
   AVP: l=6  t=Framed-MTU(12): 1500
   AVP: l=19  t=Called-Station-Id(30): AA-BB-CC-00-64-00
   AVP: l=19  t=Calling-Station-Id(31): 08-00-27-6E-C5-50
   AVP: l=202  t=EAP-Message(79) Last Segment[1]
   AVP: l=18  t=Message-Authenticator(80): 01418d3b1865556918269d3cf73608b0
```

1. 按一下右鍵**Radius Protocol**，然後選擇**Export selected packet bytes**。

2. 將RADIUS負載寫入檔案（二進位制資料）。

3. 要計算Message-Authenticator欄位，必須在該欄位中放置零並計算HMAC-MD5。

   例如，當您在鍵入「：%!xxd」之後使用十六進位制/二進位制編輯器（如vim）時，該編輯器將切換到十六進位制模式，從「5012」之後開始將零設定為16個位元組(50hex在十進位制中為80，該十進位制是消息驗證器型別，12是包含屬性值對(AVP)標頭的大小為18):

```
0000000: 0116 0167 bed9 5259 5783 02c0 f918 4df6  ...g..RYW.....M.
0000010: 2b85 9d6b 0107 6369 7363 6f06 0600 0000  +..k..cisco.....
0000020: 020c 0600 0005 dc1e 1341 412d 4242 2d43  .........AA-BB-C
0000030: 432d 3030 2d36 342d 3030 1f13 3038 2d30  C-00-64-00..08-0
0000040: 302d 3237 2d36 452d 4335 2d35 304f ca02  0-27-6E-C5-500..
0000050: 4100 c819 8000 0000 be16 0301 0086 1000  A...............
0000060: 0082 0080 880d 0fe6 8421 562e bcf3 75a7  .........!V...u.
0000070: fbf4 9c20 e114 a19d 1282 96d7 45b8 9c26  ... ........E..&
0000080: 86c5 9935 1b2c ca98 1b60 5e91 1c63 d123  ...5.,...`^..c.#
0000090: f019 1ab6 7e2d 0497 1e02 0768 0ac3 aa84  ....~-.....h....
00000a0: 80d5 cd14 92a9 ae31 e9e2 121e 28e8 5f21  .......1....(._!
00000b0: 5c1a 4e20 013f a55b 7b1d 0eb7 1d17 a565  \.N .?.[{......e
00000c0: 626b 2bb4 f756 da05 b51b 043b 346a c51f  bk+..V.....;4j..
00000d0: 98a7 007e ed55 e24b 1cab ec06 799b aed5  ...~.U.K....y...
00000e0: 72c5 451b 1403 0100 0101 1603 0100 28e2  r.E...........(.
00000f0: d25f 2deb 0f0c baf5 570d d3f6 05df 6534  ._-.....W.....e4
0000100: 48d8 0853 00ae 3230 73a9 afb7 ac87 d834  H..S..20s......4
0000110: f7e9 bb57 8ac1 1750 1200 0000 0000 0000  ...W...P........
0000120: 0000 0000 0000 0000 003d 0600 0000 0f05  .........=......
0000130: 0600 00c3 5057 0d45 7468 6572 6e65 7430  ....PW.Ethernet0
0000140: 2f30 181f 3236 5365 7373 696f 6e49 443d  /0..26SessionID=
0000150: 6163 732f 3134 3531 3136 3739 372f 3132  acs/145116797/12
0000160: 3b04 06c0 a80a 0a                        ;......
```

   在修改之後，負載準備就緒。必須返回到十六進位制/二進位制模式(型別：":%!xxd -r")並儲存檔案(":wq")。

4. 使用OpenSSL來計算HMAC-MD5:

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

HMAD-MD5函式有兩個引數：標準輸入(stdin)中的第一個是消息本身，第二個是共用金鑰（本例中為Cisco）。 結果與附加到RADIUS訪問請求資料包的消息驗證器的值完全相同。

使用Python指令碼可以計算相同內容：

```
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
```

上例展示如何從Access-Request計算Message-Authenticator欄位。對於Access-Challenge、Access-Accept和Access-Reject，邏輯完全相同，但必須記住應使用Request Authenticator，這在前一個Access-Request資料包中提供。

相關資訊

- RFC 2865
- RFC 2866
- RFC 3579
- 技術支援與文件 - Cisco Systems