RADIUS无效身份验证器和消息身份验证器故障排除指南

目录

[简介](#)
[身份验证器报头](#)
[响应的身份验证](#)
[何时应该预期验证失败？](#)
[密码隐藏](#)
[重新传输](#)
[记账](#)
[消息验证器属性](#)
[何时应使用消息验证器？](#)
[何时应该预期验证失败？](#)
[验证消息验证器属性](#)
[相关信息](#)

简介

本文档介绍两种RADIUS安全机制：

- 身份验证器报头
- 消息验证器属性

本文档介绍这些安全机制是什么、它们的使用方式，以及您预计何时会出现验证失败。

身份验证器报头

根据RFC 2865，身份验证器报头长度为16字节。在访问请求中使用时，它称为请求验证器。当它用于任何类型的响应时，它称为响应验证器。它用于：

- 响应的身份验证
- 密码隐藏

响应的身份验证

如果服务器使用正确的响应身份验证器进行响应，客户端可以计算该响应是否与有效请求相关。

客户端发送带有随机身份验证器报头的请求。然后，发送响应的服务器会使用请求数据包和共享密钥计算响应身份验证器：

```
ResponseAuth = MD5(Code + ID + Length + RequestAuth + Attributes + Secret)
```
接收响应的客户端执行相同的操作。如果结果相同，则数据包正确。

注意：知道密钥值的攻击者无法欺骗响应，除非它能够嗅探到请求。

何时应该预期验证失败？

如果交换机不再缓存请求（例如，由于超时），则会发生验证失败。 当共享密钥无效（是 — Access-Reject也包含此报头）时，您也可能会遇到此问题。 这样，网络接入设备(NAD)可以检测共享密钥不匹配。通常，身份验证、授权和记帐(AAA)客户端/服务器会将其报告为共享密钥不匹配，但不会显示详细信息。

密码隐藏

身份验证器报头也用于避免以纯文本形式发送用户密码属性。首先计算消息摘要5（MD5 — 加密，验证器）。然后，对密码块执行多个XOR操作。由于MD5不再被视为强单向算法，因此此方法很容易受到离线攻击（彩虹表）。

以下是计算用户密码的Python脚本：

```
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]))
```

重新传输

如果RADIUS访问请求中的任何属性已更改（如RADIUS ID、用户名等），应生成新的身份验证器字段，并重新计算所有依赖它的其他字段。如果这是重新传输，则不应有任何变化。

记账

身份验证器报头对于访问请求和记帐请求的含义不同。

对于访问请求，身份验证器是随机生成的，预期会收到响应，响应身份验证器计算正确，这证明响应与该特定请求相关。

对于记帐请求，身份验证器不是随机的，但是它是按照RFC 2866计算的：

```
RequestAuth = MD5(Code + ID + Length + 16 zero octets + Attributes + Secret)
```
这样，如果重新计算的值与验证器值不匹配，服务器可以立即检查记帐消息并丢弃数据包。身份服务引擎(ISE)返回：

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

出现这种情况的典型原因是共享密钥不正确。

消息验证器属性

消息验证器属性是RFC 3579中定义的RADIUS属性。其用途类似：签名和验证。但这次，它不用于验证响应，而用于验证请求。

发送访问请求的客户端（也可以是以访问质询作出响应的服务器）从自己的数据包计算基于哈希的消息身份验证代码(HMAC)-MD5，然后添加消息身份验证器属性作为签名。然后，服务器可以验证它执行了相同的操作。

公式类似于验证器报头：

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

HMAC-MD5函数包含两个参数：

- 数据包的负载，包括16字节的Message-Authenticator字段，填入零
- 共享密钥

何时应使用消息验证器？

每个数据包必须使用消息验证器，包括可扩展身份验证协议(EAP)消息(RFC 3579)。 这包括发送Access-Request的客户端和以Access-Challenge响应的服务器。如果验证失败，另一端应以静默方式丢弃数据包。

何时应该预期验证失败？

当共享密钥无效时，验证失败。然后，AAA服务器无法验证请求。

ISE报告：

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

这通常发生在EAP消息附加的后期。802.1x会话的第一个RADIUS数据包不包含EAP消息；没有消息验证器字段，无法验证请求，但在该阶段，客户端能够使用验证器字段验证响应。

验证消息验证器属性

以下示例说明如何手动计数值，以确保其计算正确。

已选择数据包编号30（访问请求）。它处于EAP会话中，且数据包包含Message-Authenticator字段。目的是验证消息验证器是否正确：

```
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. 右键单击Radius **Protocol并**选择Export **Selected packet bytes**。

2. 将该RADIUS负载写入文件（二进制数据）。

3. 要计算消息验证器字段，必须将零置于此处并计算HMAC-MD5。

   例如，当您使用十六进制/二进制编辑器（如vim）时，在键入“：%!xxd”后，它会切换到十六进制模式，从“5012”后开始为16个字节(50hex是80，其中是消息验证器类型，12是18，包括属性值对(AVP)报头):

```
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                        ;......
```

   修改后，负载就绪。必须返回十六进制/二进制模式(类型：":%!xxd -r")并保存文件(":wq")。

4. 使用OpenSSL计算HMAC-MD5:

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

HMAD-MD5函数采用两个参数：第一个来自标准输入(stdin)是消息本身，第二个是共享密钥（本例中为思科）。 结果与附加到RADIUS Access-Request数据包的消息验证器的值完全相同。

使用Python脚本可以计算相同值：

```
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
```

上一个示例介绍如何从Access-Request计算Message-Authenticator字段。对于Access-Challenge、Access-Accept和Access-Reject，逻辑完全相同，但务必记住应使用请求身份验证器，这在以前的Access-Request数据包中提供。

相关信息

- RFC 2865
- RFC 2866
- RFC 3579
- 技术支持和文档 - Cisco Systems