RADIUS无效验证器和消息验证器排除故障指南

目录

简介

本文描述两RADIUS安全机制：

- 验证器报头
- 消息验证器属性

本文包括什么这些安全机制是，如何使用他们，并且，当您应该等验证失败时。

验证器报头

每RFC 2865，验证器报头是长16个的字节。当用于Access-Request时，它呼叫请求验证器。当用于任何答复时，它呼叫答复验证器。它使用为：

- 答复的验证
- 密码隐藏

答复的验证

如果服务器回应正确答复验证器，客户端能计算，如果该答复与一有效请求涉及。

客户端发送与随机的验证器报头的请求。然后，发送答复的服务器与共享机密一起计算与使用的答复验证器请求包：

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

收到答复的客户端执行同一操作。如果结果是相同的，数据包正确。

注意：认识加密值的攻击者不能伪装答复，除非能探测请求。

什么时候应该等验证失败？

验证失败发生，如果交换机不再缓存请求(例如，由于超时)。您也许也体验它，当共享机密无效时(是-访问拒绝也包括此报头)。这样，网络接入设备(纳季)能检测共享秘密不匹配。通常它由验证、授权和统计(AAA)客户端/服务器报告作为共享密钥不匹配，但是不透露详细信息。

密码隐藏

验证器报头也用于为了避免发送在纯文本的用户密码属性。首先(MD5 -机密，验证器)计算消息摘要5。然后与密码的大块的几XOR操作被执行。此方法为脱机攻击(彩虹表)是易受，因为MD5不再被察觉作为一种强单程算法。

这是计算用户密码的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

如果其中任一个在RADIUS Access-Request的属性更改(类似RADIUS ID，用户名，等等)，应该生成新的验证器字段，并且取决于它的其他字段应该重新计算。如果这是重新传输，什么都不应该更改。

核算

验证器报头的含义为Access-Request和记帐请求是不同的。

对于Access-Request，验证器随机地生成，并且预计收到与正确地计算的ResponseAuthenticator的答复，证明，答复与该特定请求涉及。

对于记帐请求，验证器不随机，但是计算(根据RFC 2866) ：

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

如果重新计算的值不匹配验证器值，这样，服务器能立即检查核算消息和丢弃数据包。身份服务引擎(ISE)回归：

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

此的典型原因是不正确共享密钥。

消息验证器属性

消息验证器属性是在RFC 3579定义的RADIUS属性。它使用一个相似的目的：签字和验证。但是这次，它没有用于为了验证答复，然而请求。

发送Access-Request的客户端(可以也是服务器回应访问-查询)的它计算基于HASH算法的消息认证代码(从其自己的数据包的HMAC)-MD5，然后添加消息验证器属性作为签名。然后，服务器能验证它执行同一操作。

公式看起来类似于验证器报头：

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

HMAC-MD5功能在两个参数采取：

- 数据包的有效负载，包括16个字节消息验证器字段用零填装了
- 共享密钥

什么时候应该使用消息验证器？

必须用于消息验证器每数据包，包括可扩展的认证协议(EAP)消息(RFC 3579)。这包括发送Access-Request和服务器回应访问-查询的客户端。如果验证发生故障，另一侧应该静静地丢弃数据包。

什么时候应该等验证失败？

当共享机密无效，验证失败将发生。然后， AAA服务器不能验证请求。

ISE报告：

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

当EAP信息附加时，这通常发生在后期阶段。802.1x会话的第一个RADIUS信息包不包括EAP信息;没有消息验证器字段，并且验证请求是不可能的，但是在该阶段，客户端能验证与使用的答复验证器字段。

验证消息验证器属性

这是说明您如何的示例手工计数值为了确保它正确地被计算。

数据包编号30 (Access-Request)选定的。它是在EAP会话中间，并且数据包包括消息验证器字段。AIM将验证消息验证器正确：

```
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. 用鼠标右键单击**RADIUS协议**并且选择**出口所选数据包字节**。

2. 写该RADIUS有效负载到文件(二进制数据)。

3. 为了计算消息验证器字段，您必须放置零那里和计算HMAC-MD5。

   例如，当您使用六角形的/binary编辑器，例如精力，在您键入“后：%! xxd”，换成六角形的模式和零开始16个的字节，在"5012"后(50hex是80在是消息验证器类型的dec中，并且12是18包括属性值对的大小(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                        ;......
```

   以后该修改，有效负载准备好。是必要的返回返回六角形的/binary模式(类型：“：%! xxd -r”)和保存文件("：wq”)。

4. 请使用Openssl为了计算HMAC-MD5 ：

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

HMAD-MD5功能采取两个参数：第一个从标准的输入(stdin)是消息和第二个是共享机密(在本例中的思科)。结果是同一个值作为消息验证器附加到RADIUS访问请求信息包。

同样可以计算与使用Python脚本：

```
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
```

前一个示例提交如何计算从Access-Request的消息验证器字段。对于访问-查询、Access-Accept和访问拒绝，逻辑正确地是相同的，但是请记住应该使用请求验证器，在上一个访问请求信息包提供。

相关信息

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