

# O autenticador e o Mensagem-autenticador inválidos do RAIO pesquisam defeitos o guia

## Índice

[Introdução](#)

[Encabeçamento do autenticador](#)

[Autenticação da resposta](#)

[Quando deve você esperar a falha da validação?](#)

[Esconder da senha](#)

[Retransmissões](#)

[Relatório](#)

[Atributo do Mensagem-autenticador](#)

[Quando deve o Mensagem-autenticador ser usado?](#)

[Quando deve você esperar a falha da validação?](#)

[Valide o atributo do Mensagem-autenticador](#)

[Informações Relacionadas](#)

## Introdução

Este documento descreve dois mecanismos de segurança do RAIO:

- Encabeçamento do autenticador
- Atributo do Mensagem-autenticador

Este capas de documento o que estes mecanismos de segurança são, como estão usados, e quando você dever esperar a falha da validação.

## Encabeçamento do autenticador

Pelo RFC 2865, o encabeçamento do autenticador é 16 bytes por muito tempo. Quando é usado em uma solicitação de acesso, está chamado um autenticador do pedido. Quando é usado em qualquer tipo da resposta, está chamado um autenticador da resposta. É usado para:

- Autenticação da resposta
- Esconder da senha

## Autenticação da resposta

Se o server responde com o autenticador correto da resposta, o cliente pode computar se essa resposta foi relacionada a um pedido válido.

O cliente envia o pedido com o encabeçamento aleatório do autenticador. Então, o server que envia a resposta calcula o autenticador da resposta com o uso do pacote de requisição junto com o segredo compartilhado:

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

O cliente que recebe a resposta executa a mesma operação. Se o resultado é o mesmo, o pacote está correto.

Nota: O atacante que conhece o valor secreto não pode ao spoof a resposta a menos que puder aspirar o pedido.

## Quando deve você esperar a falha da validação?

A falha da validação ocorre se o interruptor não põe em esconderijo o pedido anymore (por exemplo, devido ao intervalo). Você pôde igualmente experimentá-la quando o segredo compartilhado é inválido (sim - a Rejeição de acesso igualmente inclui este encabeçamento). Esta maneira, o dispositivo do acesso de rede (NAD) pode detectar a má combinação secreta compartilhada. É relatada geralmente por clientes/server do Authentication, Authorization, and Accounting (AAA) como uma má combinação de chave compartilhada, mas não revela os detalhes.

## Esconder da senha

O encabeçamento do autenticador é usado igualmente a fim evitar enviar o atributo da Senha do usuário no texto simples. O message digest 5 (MD5 - segredo, autenticador) é computado primeiramente. Diversas operações XOR com os pedaços da senha são executadas então. Este método é susceptível para ataques autônomos (tabelas do arco-íris) porque o MD5 não é percebido como um algoritmo de sentido único forte anymore.

Está aqui o script do pitão que computa a Senha do usuário:

```
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]))
```

## Retransmissões

Se alguns dos atributos na solicitação de acesso do RAIIO mudaram (como o RAIIO ID, username, e assim por diante), o campo novo do autenticador deve ser gerado e todos campos restantes que dependem dele devem ser recalculados. Se esta é uma retransmissão, nada deve mudar.

## Relatório

O significado do encabeçamento do autenticador é diferente para uma solicitação de acesso e um Contabilidade-pedido.

Para uma solicitação de acesso, o autenticador é gerado aleatoriamente e espera-se receber uma resposta com o ResponseAuthenticator calculado corretamente, que mostra que a resposta esteve relacionada a esse pedido específico.

Para um Contabilidade-pedido, o autenticador não é aleatório, mas calcula-se (conforme o RFC 2866):

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

Esta maneira, o server pode verificar a mensagem da contabilidade imediatamente e deixar cair o pacote se o valor voltado a calcular não combina o valor do autenticador. Os retornos do Identity Services Engine (ISE):

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

O motivo típico para este é a chave secreta compartilhada incorreta.

## Atributo do Mensagem-autenticador

O atributo do Mensagem-autenticador é o atributo RADIUS definido no RFC 3579. É usado para uma finalidade similar: para assinar e validar. Mas esta vez, não é usado a fim validar uma resposta mas um pedido.

O cliente que envia uma solicitação de acesso (ele pode igualmente ser um server que responda com um Acesso-desafio) computa o Hash-Based Message Authentication Code (HMAC)-MD5 de seu próprio pacote, e adiciona então o atributo do Mensagem-autenticador como uma assinatura. Então, o server pode verificar que executa a mesma operação.

A fórmula olha similar ao encabeçamento do autenticador:

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

A função HMAC-MD5 recolhe dois argumentos:

- O payload do pacote, que inclui o campo de um Mensagem-autenticador de 16 bytes encheu-se com os zero
- O segredo compartilhado

## Quando deve o Mensagem-autenticador ser usado?

O Mensagem-autenticador DEVE ser usado para cada pacote, que inclui a mensagem do Extensible Authentication Protocol (EAP) (RFC 3579). Isto inclui o cliente que envia a solicitação de acesso e o server que responde com o Acesso-desafio. O outro lado deve silenciosamente deixar cair o pacote se a validação falha.

## Quando deve você esperar a falha da validação?

A falha da validação ocorrerá quando o segredo compartilhado é inválido. Então, o servidor AAA não pode validar o pedido.

Os relatórios ISE:

11036 The Message-Authenticator Radius Attribute is invalid.

Isto ocorre geralmente no estágio posterior quando o mensagem EAP é anexado. O primeiro pacote de informação de RADIUS da sessão do 802.1x não inclui o mensagem EAP; não há nenhum campo do Mensagem-autenticador e não é possível verificar o pedido, mas nessa fase, o cliente pode validar a resposta com o uso do campo do autenticador.

## Valide o atributo do Mensagem-autenticador

Está aqui um exemplo para ilustrar como você conta manualmente o valor a fim se certificar que está computado corretamente.

O pacote número 30 (solicitação de acesso) foi escolhido. É no meio da sessão EAP, e o pacote inclui o campo do Mensagem-autenticador. O alvo é verificar que o Mensagem-autenticador está correto:

```
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): 01418d3b1865556918269d3c f73608b0
```

1. Clicar com o botão direito o **protocolo de raio** e escolha **bytes de pacote selecionados exportação**.
2. Escreva esse payload do RAI0 a um arquivo (dados binários).
3. A fim computar o campo do Mensagem-autenticador, você deve pôr zero lá e computar o HMAC-MD5.

Por exemplo, quando você se usa encantar/editor binário, tal como o vim, depois que você datilografa “: %! o xxd”, que comuta para encantar o modo e os zero 16 bytes que começam depois que “5012” (50hex é 80 em dezembro que é tipo do Mensagem-autenticador, e 12 é o tamanho que é 18 que incluem o encabeçamento dos pares de valor de atributo (AVP)):

```

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 ;.....

```

Em seguida essa alteração, o payload está pronta. É necessário retornar de volta a encanta/modo binário (tipo: “: %!” o xxd - r”) e salvar o arquivo (“: wq”).

#### 4. Use o OpenSSL a fim computar o HMAC-MD5:

```

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

```

A função HMAD-MD5 toma dois argumentos: primeiro da entrada padrão (stdin) é a mensagem própria e segunda é o segredo compartilhado (Cisco neste exemplo). O resultado é exatamente o mesmo valor como o Mensagem-autenticador anexado ao pacote de solicitação de acesso do RAIO.

O mesmos podem ser computados com o uso do script do pitão:

```

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

```

O exemplo anterior apresenta como calcular o campo do Mensagem-autenticador da solicitação de acesso. Para o Acesso-desafio, a aceitação de acesso, e a Rejeição de acesso, a lógica é exatamente a mesma, mas é importante recordar que o autenticador do pedido deve ser usado, que é fornecido no pacote de solicitação de acesso precedente.

## Informações Relacionadas

- [RFC 2865](#)
- [RFC 2866](#)
- [RFC 3579](#)
- [Suporte Técnico e Documentação - Cisco Systems](#)