

RADIUS Ongeldige authenticator en Message-Authenticator probleemoplossing

Inhoud

[Inleiding](#)

[Verificatieheader](#)

[Verificatie van de respons](#)

[Wanneer zou u validatie-storing verwachten?](#)

[Wachtwoordblokkering](#)

[Terugzending](#)

[accounting](#)

[Kenmerk voor e-mailverificatie](#)

[Wanneer moet de Message-Authenticator worden gebruikt?](#)

[Wanneer zou u validatie-storing verwachten?](#)

[Verificatiekenmerk](#)

[Gerelateerde informatie](#)

Inleiding

In dit document worden twee RADIUS-beveiligingsmechanismen beschreven:

- Verificatieheader
- Attribution-kenmerk

Dit document beschrijft wat deze beveiligingsmechanismen zijn, hoe ze worden gebruikt en wanneer u een validatiestoornis zou moeten verwachten.

Verificatieheader

Per RFC 2865 is de header van de verificator 16 bytes lang. Wanneer het in een Toegang-Verzoek wordt gebruikt, wordt het genoemd een Authenticator van het Verzoek. Wanneer het gebruikt wordt in elke vorm van respons, wordt het een Response Authenticator genoemd. Het wordt gebruikt voor:

- Verificatie van de respons
- Wachtwoordverstopping

Verificatie van de respons

Als de server reageert met de juiste Response Authenticator, kan de client berekenen als die

respons gerelateerd was aan een geldig verzoek.

De client stuurt het verzoek door de willekeurige verificatieheader. Vervolgens berekent de server die de respons verstuurt de Response Authenticator met het gebruik van het verzoepakket samen met het gedeelde geheim:

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

De client die de reactie ontvangt, voert dezelfde handeling uit. Als het resultaat hetzelfde is, klopt het pakje.

Opmerking: De aanvaller die de geheime waarde kent kan de reactie niet bederven tenzij hij in staat is om het verzoek te besnuffelen.

Wanneer zou u validatie-storing verwachten?

Validering treedt op als de switch het verzoek niet meer in het geheugen stelt (bijvoorbeeld vanwege de onderbreking). U zou het ook kunnen ervaren wanneer het gedeelde geheim ongeldig is (ja - Access-Afwijzen omvat ook deze header). Op deze manier kan het Network Access Apparaat (NAD) de gedeelde geheime tegenstrijdigheid detecteren. Meestal wordt dit door AAA-klienten/servers (Verificatie, autorisatie en accounting) gemeld als een gedeelde belangrijke mismatch, maar de details worden niet onthuld.

Wachtwoordblokkering

De header van de verficator wordt ook gebruikt om te voorkomen dat de gebruiker-Wachtwoord in onbewerkte tekst wordt verzonden. Eerst wordt het bericht Grootte 5 (MD5 - geheim, authentiek) berekend. Vervolgens worden verscheidene XOR - transacties met de aantekeningen van het wachtwoord uitgevoerd. Deze methode is gevoelig voor offline aanvallen (regenboogtabellen) omdat MD5 niet meer gezien wordt als een sterk algoritme in één richting.

Dit is het Python-script dat het User-Password compileert:

```
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]))
```

Terugzending

Als een van de eigenschappen in het RADIUS-toegangsverzoek is gewijzigd (zoals de RADIUS-id, de User-Name, enzovoort), dan moet het veld nieuwe Authenticator worden gegenereerd en moeten alle andere velden die ervan afhankelijk zijn, opnieuw worden berekend. Als dit een heroverdracht is, zou niets moeten veranderen.

accounting

De betekenis van de header van de verificator is anders voor een toegangsaanvraag en een accounting-aanvraag.

Voor een toegangsaanvraag wordt de authenticator willekeurig gegenereerd en wordt verwacht dat hij een reactie ontvangt met de ResponseAuthenticator die correct berekend is, wat bewijst dat de respons gerelateerd was aan dat specifieke verzoek.

Voor een accounting-aanvraag is de Authenticator niet willekeurig, maar het wordt berekend (zoals per RFC 2866):

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

Op deze manier kan de server het accountantsbericht direct controleren en het pakket laten vallen indien de herberekende waarde niet overeenkomt met de waarde voor Authenticator. Identity Services Engine (ISE) retourneert:

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

De typische reden hiervoor is de onjuiste gedeelde geheime sleutel.

Kenmerk voor e-mailverificatie

De eigenschap Message-Authenticator is de RADIUS-eigenschap gedefinieerd in RFC 3579. Het wordt voor een soortgelijk doel gebruikt: tekenen en valideren . Maar deze keer wordt het niet gebruikt om een antwoord maar een verzoek te valideren.

De client die een toegangsaanvraag verstuurt (het kan ook een server zijn die reageert met een Access-Challenge) compileert de Hash-Based Berichtverificatie Code (HMAC)-MD5 uit zijn eigen pakket en voegt vervolgens de eigenschap Message-Authenticator toe als een handtekening. Vervolgens kan de server controleren of het dezelfde handeling uitvoert.

De formule lijkt op de header van de verificator:

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

De HMAC-MD5-functie voert twee argumenten in:

- De lading van het pakje, dat het veld 16 bytes Message-Authenticator met nullen bevat
- Het gedeelde geheim

Wanneer moet de Message-Authenticator worden gebruikt?

De Message-Authenticator moet worden gebruikt voor elk pakje, dat het MAP-bericht (Extensible Authentication Protocol) bevat (RFC 3579). Dit omvat zowel de client die het toegangsverzoek verstuurt als de server die met de toegangsuitdaging reageert. De andere zijde zou het pakket stilletjes moeten laten vallen als de validatie faalt.

Wanneer zou u validatie-storing verwachten?

De validatie zal mislukken wanneer het gedeelde geheim ongeldig is. Vervolgens kan de AAA-server het verzoek niet valideren.

De ISE meldt:

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

Dit gebeurt meestal in de latere fase, wanneer het MAP-bericht wordt bijgevoegd. Het eerste RADIUS-pakket van de 802.1x-sessie bevat niet het MAP-bericht; er is geen veld Berichtverificatie en het is niet mogelijk het verzoek te verifiëren, maar in dat stadium kan de cliënt de reactie valideren met het gebruik van het veld Verificator.

Verificatiekenmerk

Hier is een voorbeeld om aan te geven hoe u handmatig de waarde telt om ervoor te zorgen dat deze correct wordt berekend.

Packet nummer 30 (Access-Application) is geselecteerd. Het bevindt zich midden in de MAP-sessie en het pakket bevat het veld Berichtverificatie-authenticator. Het doel is te controleren of de Berichtverificatie correct is:

```
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. Klik met de rechtermuisknop op **Radius Protocol** en kies **Geselecteerde pakketbytes exporteren**.
2. Schrijf die RADIUS-lading naar een bestand (binair gegevens).
3. Om het veld Message-Authenticator te berekenen, moet u daar nullen plaatsen en de HMAC-MD5 berekenen.

Bijvoorbeeld, wanneer u hex/binair editor gebruikt, zoals vim, nadat u ":%!xxd" typt, welke switches naar hex mode en zeroes 16 bytes start na "5012" (50hex is 80 dec, wat een Message-Authenticator type is, en 12 is de grootte die 18 inclusief de Attribution Value Parks (AVP) header) is:

```

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 .?.[f.....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 ;.....

```

Na die verandering is de lading klaar. Terugkeren naar hex/binaire modus (type: "%!xxd -r") en slaat het bestand op (":wq").

4. Gebruik OpenSSL om HMAC-MD5 te berekenen:

```

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

```

De HMAD-MD5 functie voert twee argumenten aan: het eerste van standaardinput (stdin) is het bericht zelf en het tweede is het gedeelde geheim (Cisco in dit voorbeeld). Het resultaat is precies dezelfde waarde als de Message-Authenticator die aan het pakket RADIUS-toegangsaanvraag is toegevoegd.

Hetzelfde kan worden berekend met behulp van het Python-schrift:

```

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
```

Het vorige voorbeeld geeft aan hoe het veld Berichtverificatie uit de toegangsaanvraag moet worden berekend. Voor Access-Challenge, Access-Accept en Access-Afwijst is de logica precies hetzelfde, maar het is belangrijk om te onthouden dat Application Authenticator moet worden gebruikt, wat in het vorige Access-Application pakket is voorzien.

Gerelateerde informatie

- [RFC 2865](#)
- [RFC 2866](#)
- [RFC 3579](#)
- [Technische ondersteuning en documentatie – Cisco Systems](#)