

RADIUS недопустимое руководство устранения неполадок средства проверки подлинности и средства проверки подлинности сообщения

Содержание

[Введение](#)

[Заголовок средства проверки подлинности](#)

[Аутентификация ответа](#)

[Когда необходимо ожидать сбоя проверки?](#)

[Соккрытие пароля](#)

[Повторные передачи](#)

[Учет](#)

[Атрибут средства проверки подлинности сообщения](#)

[Когда должно использоваться Средство проверки подлинности сообщения?](#)

[Когда необходимо ожидать сбоя проверки?](#)

[Проверьте атрибут средства проверки подлинности сообщения](#)

[Дополнительные сведения](#)

Введение

Этот документ описывает два механизма обеспечения безопасности RADIUS:

- Заголовок средства проверки подлинности
- Атрибут средства проверки подлинности сообщения

Этот документ покрывает, каковы эти механизмы обеспечения безопасности, как они используются, и когда необходимо ожидать сбоя проверки.

Заголовок средства проверки подлинности

На RFC 2865 Заголовок Средства проверки подлинности 16 байтов длиной. Когда это используется в Access-Request, это называют Средством проверки подлинности Запроса. Когда это используется в любом виде ответа, это называют Средством проверки подлинности Ответа. Это используется для:

- Аутентификация ответа
- Соккрытие пароля

Аутентификация ответа

Если сервер отвечает корректным Средством проверки подлинности Ответа, клиент может вычислить, если тот ответ был отнесен к допустимому запросу.

Клиент отправляет запрос со случайным Заголовком Средства проверки подлинности. Затем сервер, который передает ответ, вычисляет Средство проверки подлинности Ответа с использованием пакета запроса наряду с общим секретным ключом:

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

Клиент, который получает ответ, выполняет ту же операцию. Если результатом является то же, пакет корректен.

Примечание: Атакующий, который знает секретное значение, не в состоянии имитировать ответ, пока это не в состоянии осуществить sniffing запроса.

Когда необходимо ожидать сбоя проверки?

Если коммутатор больше не кэширует запрос (например, из-за таймаута), сбой проверки происходит. Вы могли бы также испытать его, когда общий секретный ключ недопустим (да - Access-Reject также включает этот заголовок). Таким образом, Устройство доступа к сети (NAD) может обнаружить несоответствие общего секретного ключа. Обычно об этом сообщают клиенты/серверы Аутентификации, авторизации и учета (AAA) как несоответствие общего ключа, но это не показывает подробные данные.

Соккрытие пароля

Заголовок Средства проверки подлинности также используется во избежание передачи атрибута User-Password в открытом тексте. Сначала Профиль сообщения 5 (MD5 - тайна, средство проверки подлинности) вычислен. Затем несколько операций XOR с блоками пароля выполняются. Этот метод восприимчив для офлайновых атак (таблицы радуги), потому что MD5 больше не воспринимается как сильный односторонний алгоритм.

Вот сценарий Python, который вычисляет User-Password:

```
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]))
```

Повторные передачи

Если какой-либо из атрибутов в Access-Request RADIUS изменился (как ID RADIUS, Имя пользователя, и так далее), новое поле Authenticator должно генерироваться и все другие поля, которые зависят от него, должен быть повторно вычислен. Если это - повторная передача, ничто не должно изменяться.

Учет

Значение Заголовка Средства проверки подлинности является другим для Access-Request и Бухгалтерского Запроса.

Для Access-Request Средство проверки подлинности генерируется случайным образом, и оно, как ожидают, получит ответ с ResponseAuthenticator, вычисленным правильно, который доказывает, что ответ был отнесен к тому конкретному запросу.

Для Бухгалтерского Запроса Средство проверки подлинности не случайно, но оно вычислено (согласно RFC 2866):

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

Если перерасчетное значение не совпадает со значением Средства проверки подлинности, Таким образом, сервер может сразу проверить бухгалтерское сообщение и отбросить пакет. Платформа Identity Services Engine (ISE) возвращается:

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

Типичная причина для этого является неправильным общим секретным ключом.

Атрибут средства проверки подлинности сообщения

Атрибут Средства проверки подлинности сообщения является атрибутом RADIUS, определенным в RFC 3579. Это используется для подобной цели: подписать и проверить. Но на этот раз, это не используется для проверки ответа, но запроса.

Клиент, который передает Access-Request (это может также быть сервер, который отвечает проблемой Доступа) вычисляет Hash-Based Message Authentication Code (HMAC) - MD5 от его собственного пакета, и затем добавляет атрибут Средства проверки подлинности сообщения как подпись. Затем сервер в состоянии проверить, что он выполняет ту же операцию.

Формула выглядит подобной Заголовку Средства проверки подлинности:

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

Функция HMAC-MD5 берет в двух аргументах:

- Информационное наполнение пакета, который включает 16-байтовое поле Message-Authenticator, заполненное нулями
- Общий секрет

Когда должно использоваться Средство проверки подлинности сообщения?

MUST Средства проверки подлинности сообщения использоваться для каждого пакета, который включает сообщение Протокола EAP (RFC 3579). Это включает и клиента, который передает Access-Request и сервер, который отвечает проблемой Доступа. Если проверка отказывает, другая сторона должна тихо отбросить пакет.

Когда необходимо ожидать сбоя проверки?

Когда общий секретный ключ будет недопустим, сбой проверки произойдет. Затем AAA-сервер не в состоянии проверить запрос.

Отчёты о ISE:

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

Когда сообщение EAP подключено, это обычно происходит в последующем этапе. Первый Пакет RADIUS сеанса 802.1x не включает сообщение EAP; нет никакого поля Message-Authenticator, и не возможно проверить запрос, но на том этапе, клиент в состоянии проверить ответ с использованием поля Authenticator.

Проверьте атрибут средства проверки подлинности сообщения

Вот пример, чтобы проиллюстрировать, как вы вручную считываете значение, чтобы удостовериться, что это вычислено правильно.

Пакет номер 30 (Access-Request) был выбран. Это посреди сеанса EAP, и пакет включает поле Message-Authenticator. Цель состоит в том, чтобы проверить, что Средство проверки подлинности сообщения корректно:

```
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 311]
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. Щелкните правой кнопкой мыши **Протокол RADIUS** и выберите, **Export выbral байты пакета**.
2. Запишите то информационное наполнение RADIUS в файл (двоичные данные).
3. Для вычислений поля Message-Authenticator необходимо поместить нули там и вычислить HMAC-MD5.

Например, при использовании hex/Бинарного редактора, такого как энергия после ввода ": %! xhd", который переключается для преобразовывания режима в шестнадцатеричную систему и обнуляет 16 байтов, запускающихся после "5012" (50hex 80 в декабре, который является типом Средства проверки подлинности сообщения, и 12, является размером, который является 18 включая заголовок Пар значений атрибутов (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 ;.....

```

После той модификации информационное наполнение готово. Необходимо вернуться назад к hex/бинарному режиму (тип: ": %! xxd-r"), и сохранили файл (": wq").

4. Используйте OpenSSL для вычислений HMAC-MD5:

```

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

```

Функция HMAc-MD5 берет два аргумента: первый от стандартного ввода (stdin) является самим сообщением, и второй является общим секретным ключом (Cisco в данном примере). Результатом является точно то же значение как Средство проверки подлинности сообщения, подключенное к Пакету запроса доступа RADIUS.

То же может быть вычислено с использованием сценария Python:

```

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
```

Предыдущий пример представляет, как вычислить поле Message-Authenticator от Access-Request. Для проблемы Доступа, Access-Accept и Access-Reject, логика является точно тем же, но важно помнить, что Средство проверки подлинности Запроса должно использоваться, который предоставлен в предыдущем Пакете запроса доступа.

Дополнительные сведения

- [RFC 2865](#)
- [RFC 2866](#)
- [RFC 3579](#)
- [Cisco Systems – техническая поддержка и документация](#)