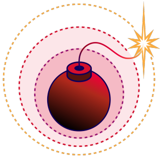
Blast-RADIUS

Breaking Enterprise Network Authentication

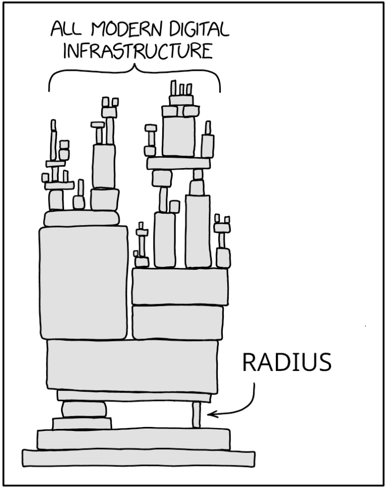
Sharon Goldberg[1], **Miro Haller**[2], Nadia Heninger[2], Mike Milano[3], Dan Shumow[4],
Marc Stevens[5], Adam Suhl[2]

[1]Cloudflare, [2]UC San Diego, [3]BastionZero, [4]Microsoft Research, [5]Centrum Wiskunde & Informatica

RealWorldCrypto; March 26, 2025

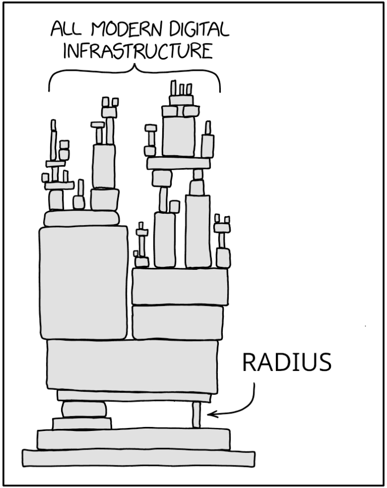


What is RADIUS? Where is it used?



modified XKCD from [7]

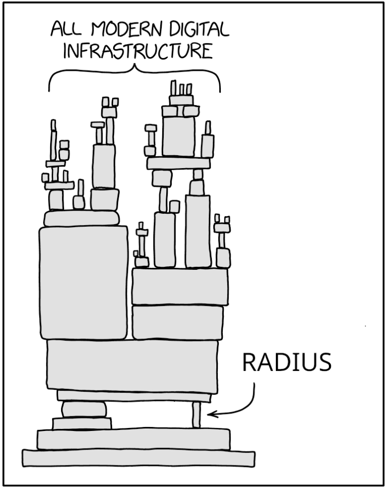
What is RADIUS? Where is it used?



modified XKCD from [7]

- *RADIUS:* standard protocol for enterprise network authentication.

What is RADIUS? Where is it used?



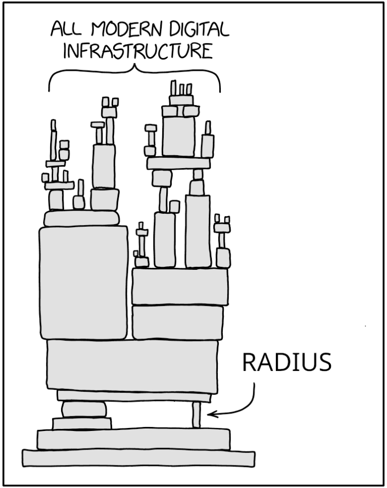
modified XKCD from [7]

- *RADIUS:* standard protocol for enterprise network authentication.
- RADIUS is *everywhere*:

  RADIUS is [...] supported by essentially every switch, router, access point, and VPN concentrator product sold in the past twenty-five years.

  (Alan DeKok [4])

What is RADIUS? Where is it used?



modified XKCD from [7]

- *RADIUS:* standard protocol for enterprise network authentication.
- RADIUS is *everywhere*:

  RADIUS is [...] supported by essentially every switch, router, access point, and VPN concentrator product sold in the past twenty-five years.

  (Alan DeKok [4])
- Used for backbone routers, non-cable ISP, IoT devices, identity providers (Okta, Duo), 802.1X, enterprise WiFi, eduroam...

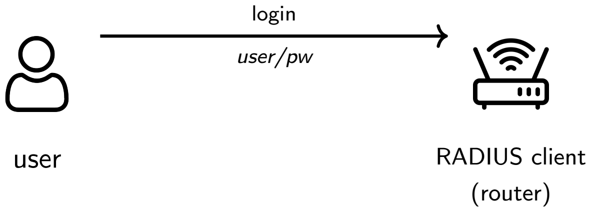
Blast-RADIUS on a Single Slide

Blast-RADIUS on a Single Slide

How does RADIUS work?

Blast-RADIUS on a Single Slide

How does RADIUS work?



login

user/pw

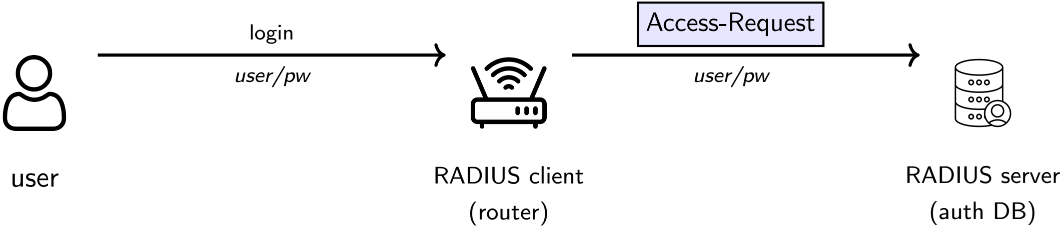
user

RADIUS client
(router)

icons from [5]

Blast-RADIUS on a Single Slide

How does RADIUS work?



login

user/pw

Access-Request

user/pw

user

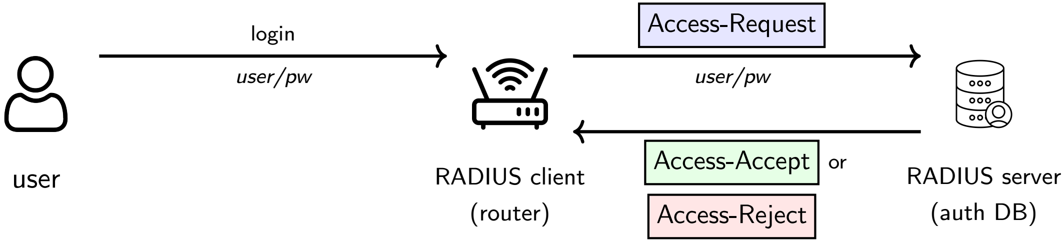
RADIUS client
(router)

RADIUS server
(auth DB)

icons from [5]

Blast-RADIUS on a Single Slide

How does RADIUS work?



login

user/pw

user

RADIUS client
(router)

Access-Request

user/pw

RADIUS server
(auth DB)

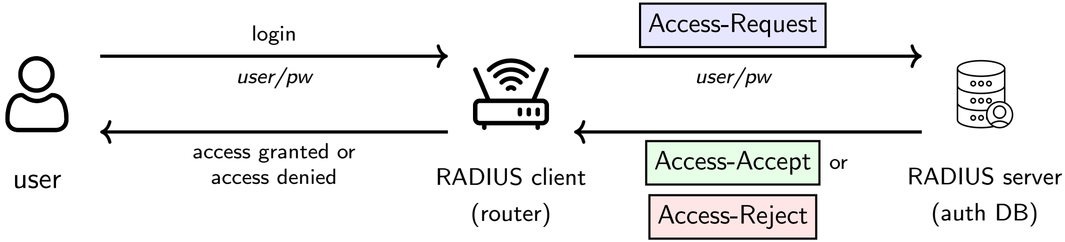
Access-Accept or

Access-Reject

icons from [5]

Blast-RADIUS on a Single Slide

How does RADIUS work?



login

user/pw

access granted or access denied

user

Access-Request

user/pw

Access-Accept or

Access-Reject

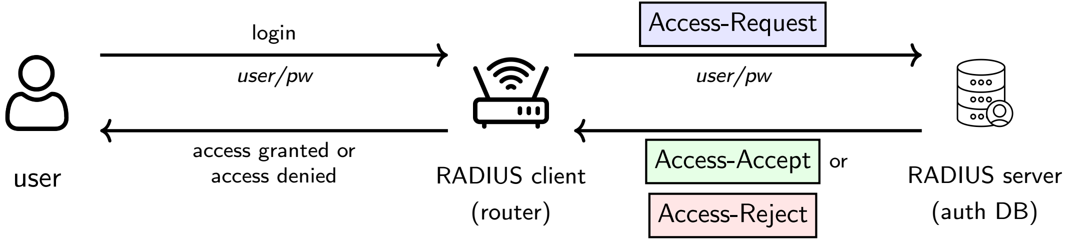
RADIUS client (router)

RADIUS server (auth DB)

icons from [5]

Blast-RADIUS on a Single Slide

How does RADIUS work?

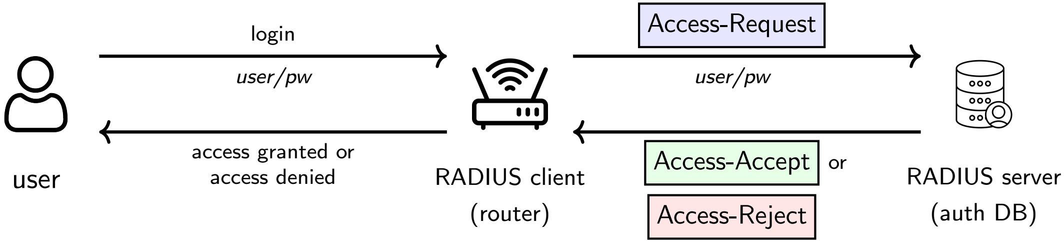


- Most RADIUS traffic is sent over UDP.

icons from [5]

Blast-RADIUS on a Single Slide

How does RADIUS work?

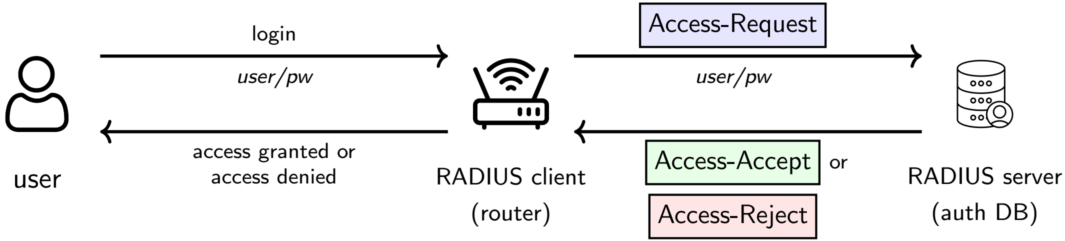


- Most RADIUS traffic is sent over UDP.
- *Our protocol vulnerability:* MITM can change Access-Reject to Access-Accept.

icons from [5]

Blast-RADIUS on a Single Slide

How does RADIUS work?

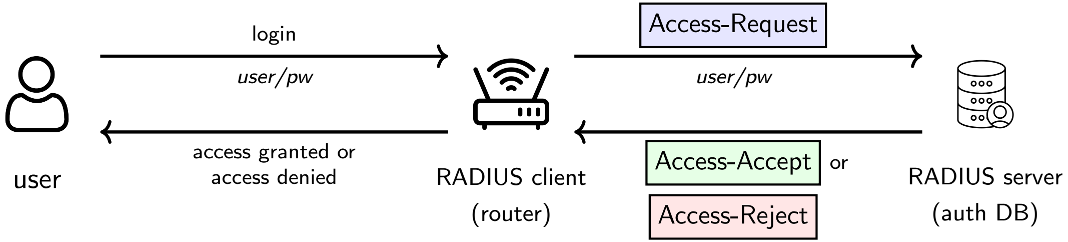


- Most RADIUS traffic is sent over UDP.
- *Our protocol vulnerability:* MITM can change Access-Reject to Access-Accept.
- *Impact:* authenticate as any user; accelerate RADIUS/UDP deprecation.

icons from [5]

Blast-RADIUS on a Single Slide

How does RADIUS work?

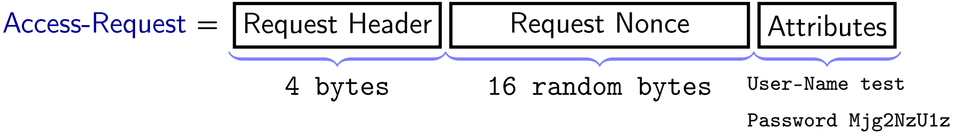


- Most RADIUS traffic is sent over UDP.
- *Our protocol vulnerability:* MITM can change Access-Reject to Access-Accept.
- *Impact:* authenticate as any user; accelerate RADIUS/UDP deprecation.
- *Mitigation:* responsible disclosure with over 90 vendors (incl. Cisco, Microsoft, ...).

icons from [5]

THE RADIUS PROTOCOL

RADIUS Packet Formats



Access-Request = Request Header | Request Nonce | Attributes

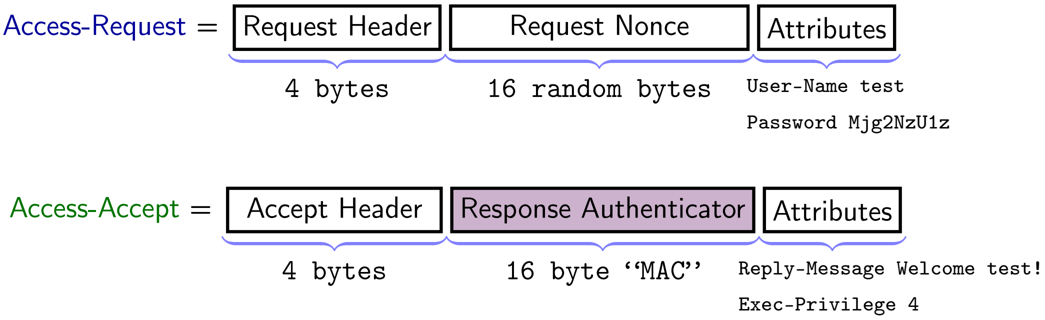
Request Header: `4 bytes`

Request Nonce: `16 random bytes`

Attributes:

```
User-Name test
Password Mjg2NzU1z
```

RADIUS Packet Formats



Access-Request = Request Header | Request Nonce | Attributes

4 bytes

16 random bytes

`User-Name test`

`Password Mjg2NzU1z`

Access-Accept = Accept Header | Response Authenticator | Attributes

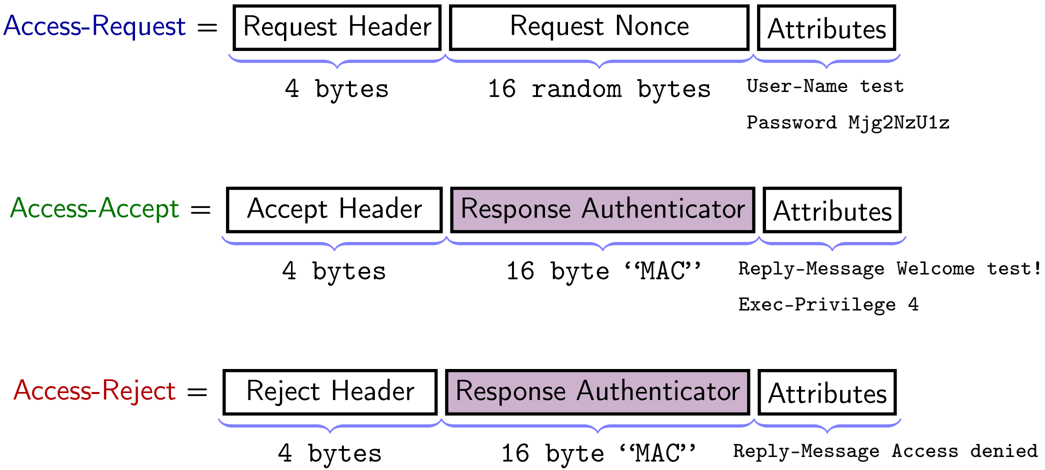
4 bytes

16 byte ‘‘MAC’’

`Reply-Message Welcome test!`

`Exec-Privilege 4`

RADIUS Packet Formats



Access-Request = Request Header | Request Nonce | Attributes

- Request Header: 4 bytes
- Request Nonce: 16 random bytes
- Attributes:

```
User-Name test
Password Mjg2NzU1z
```

Access-Accept = Accept Header | Response Authenticator | Attributes

- Accept Header: 4 bytes
- Response Authenticator: 16 byte “MAC”
- Attributes:

```
Reply-Message Welcome test!
Exec-Privilege 4
```

Access-Reject = Reject Header | Response Authenticator | Attributes

- Reject Header: 4 bytes
- Response Authenticator: 16 byte “MAC”
- Attributes:

```
Reply-Message Access denied
```

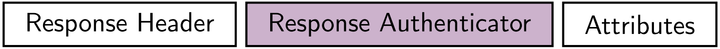
Response Authenticator

Goal: Prevent forgery of packets (e.g., by MITM attacker).

Response Authenticator

Goal: Prevent forgery of packets (e.g., by MITM attacker).

The Response Authenticator from packet

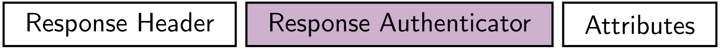


Response Header | Response Authenticator | Attributes

Response Authenticator

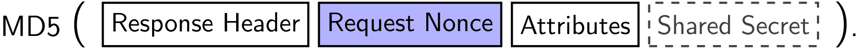
Goal: Prevent forgery of packets (e.g., by MITM attacker).

The Response Authenticator from packet



Response Header | Response Authenticator | Attributes

is computed as

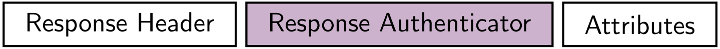


MD5 (Response Header | Request Nonce | Attributes | Shared Secret).

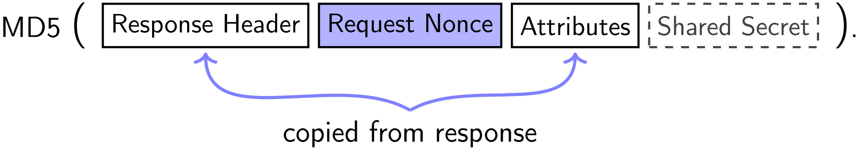
Response Authenticator

Goal: Prevent forgery of packets (e.g., by MITM attacker).

The Response Authenticator from packet


Response Header
Response Authenticator
Attributes

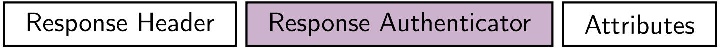
is computed as


MD5 (
Response Header
Request Nonce
Attributes
Shared Secret
).
copied from response

Response Authenticator

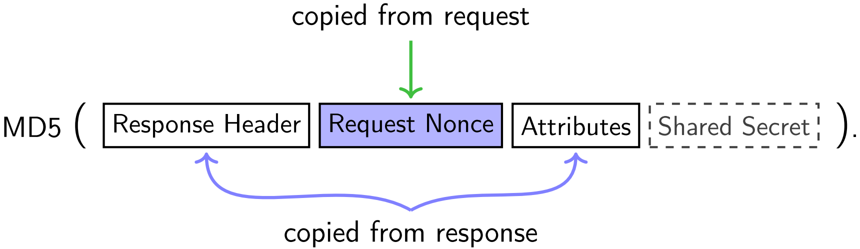
Goal: Prevent forgery of packets (e.g., by MITM attacker).

The Response Authenticator from packet



Response Header | Response Authenticator | Attributes

is computed as



copied from request

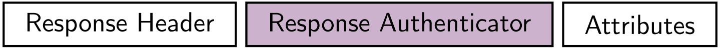
MD5 (Response Header | Request Nonce | Attributes | Shared Secret).

copied from response

Response Authenticator

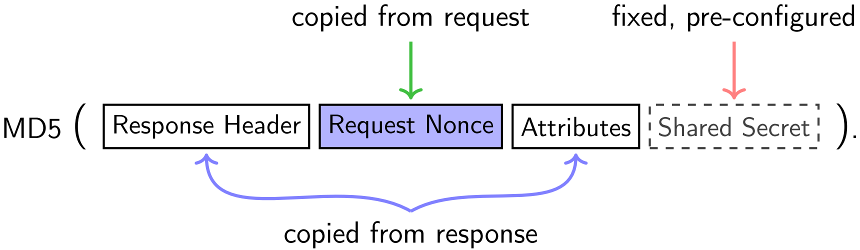
Goal: Prevent forgery of packets (e.g., by MITM attacker).

The Response Authenticator from packet



Response Header | Response Authenticator | Attributes

is computed as



copied from request

fixed, pre-configured

MD5 (Response Header | Request Nonce | Attributes | Shared Secret).

copied from response

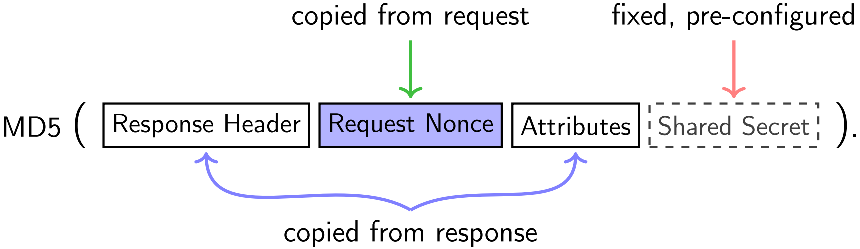
Response Authenticator

Goal: Prevent forgery of packets (e.g., by MITM attacker).

The Response Authenticator from packet

Response Header | Response Authenticator | Attributes

is computed as

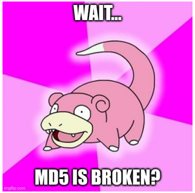


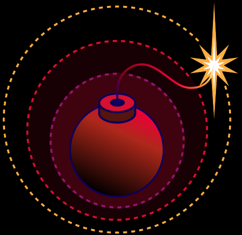
copied from request

fixed, pre-configured

MD5 (Response Header | Request Nonce | Attributes | Shared Secret).

copied from response





THE BLAST-RADIUS ATTACK

Blast-RADIUS: Attack Overview

Goal: Forge Access-Accept without knowing shared secret.

Blast-RADIUS: Attack Overview

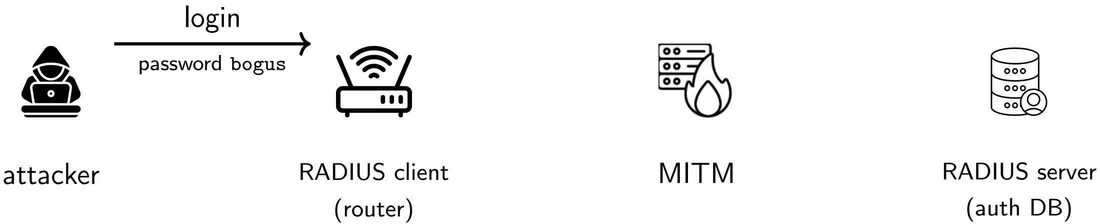
Goal: Forge Access-Accept without knowing shared secret.

Blast-RADIUS attack: Create MD5 collision s.t. Access-Accept and Access-Reject produce same Response Authenticator: MD5(Access-Accept) = MD5(Access-Reject).

Blast-RADIUS: Attack Overview

Goal: Forge Access-Accept without knowing shared secret.

Blast-RADIUS attack: Create MD5 collision s.t. Access-Accept and Access-Reject produce same Response Authenticator: MD5(Access-Accept) = MD5(Access-Reject).



login

password bogus

attacker

RADIUS client
(router)

MITM

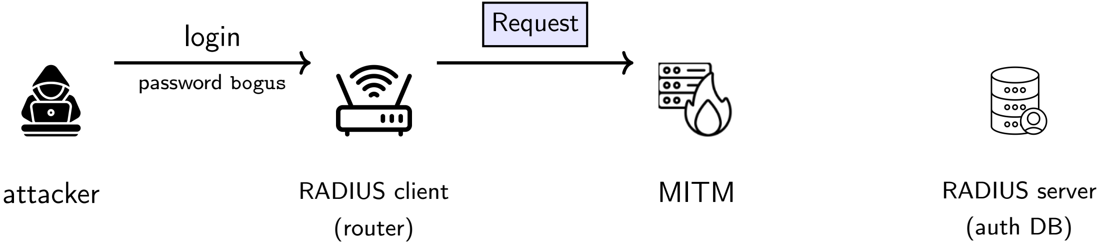
RADIUS server
(auth DB)

icons from [5]

Blast-RADIUS: Attack Overview

Goal: Forge Access-Accept without knowing shared secret.

Blast-RADIUS attack: Create MD5 collision s.t. Access-Accept and Access-Reject produce same Response Authenticator: MD5(Access-Accept) = MD5(Access-Reject).

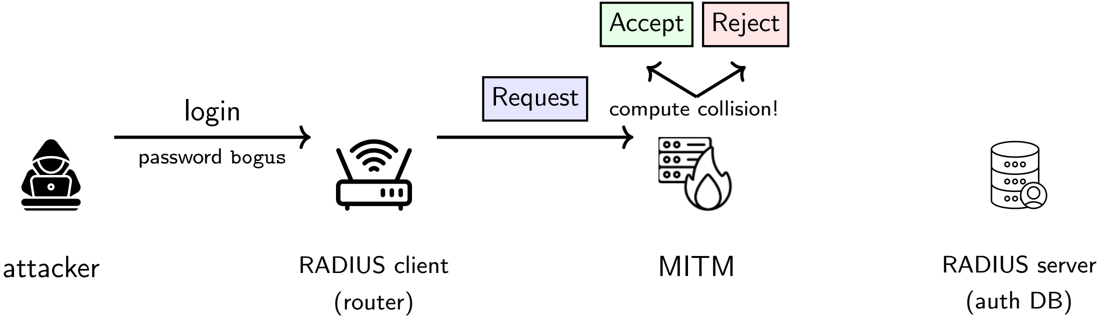


icons from [5]

Blast-RADIUS: Attack Overview

Goal: Forge Access-Accept without knowing shared secret.

Blast-RADIUS attack: Create MD5 collision s.t. Access-Accept and Access-Reject produce same Response Authenticator: MD5(Access-Accept) = MD5(Access-Reject).

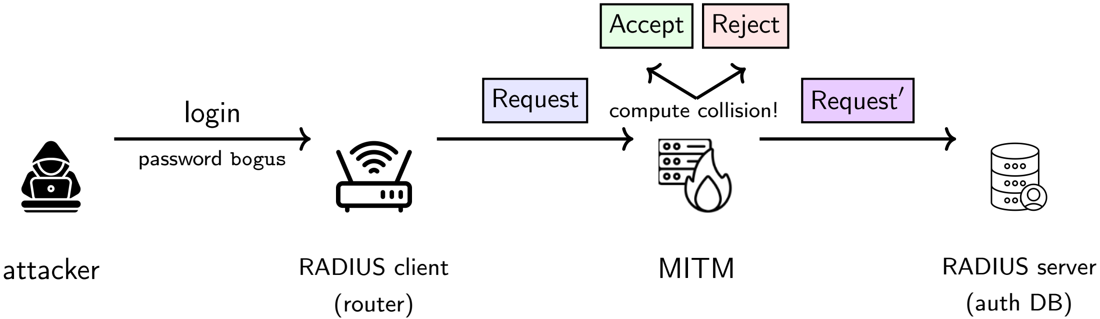


icons from [5]

Blast-RADIUS: Attack Overview

Goal: Forge Access-Accept without knowing shared secret.

Blast-RADIUS attack: Create MD5 collision s.t. Access-Accept and Access-Reject produce same Response Authenticator: MD5(Access-Accept) = MD5(Access-Reject).

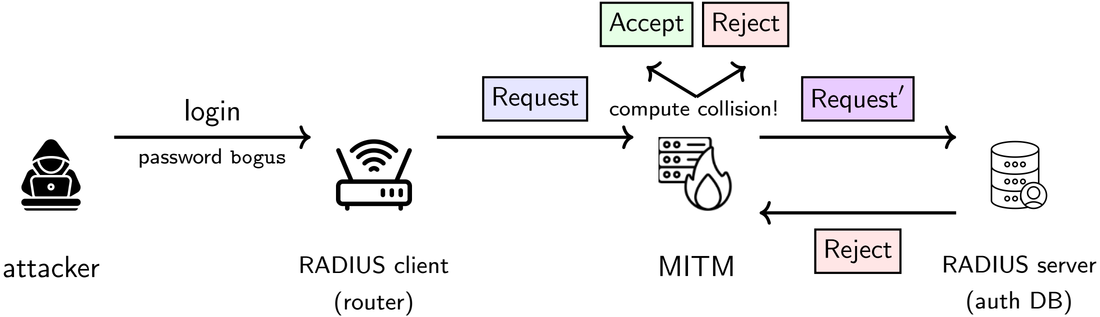


icons from [5]

Blast-RADIUS: Attack Overview

Goal: Forge Access-Accept without knowing shared secret.

Blast-RADIUS attack: Create MD5 collision s.t. Access-Accept and Access-Reject produce same Response Authenticator: MD5(Access-Accept) = MD5(Access-Reject).

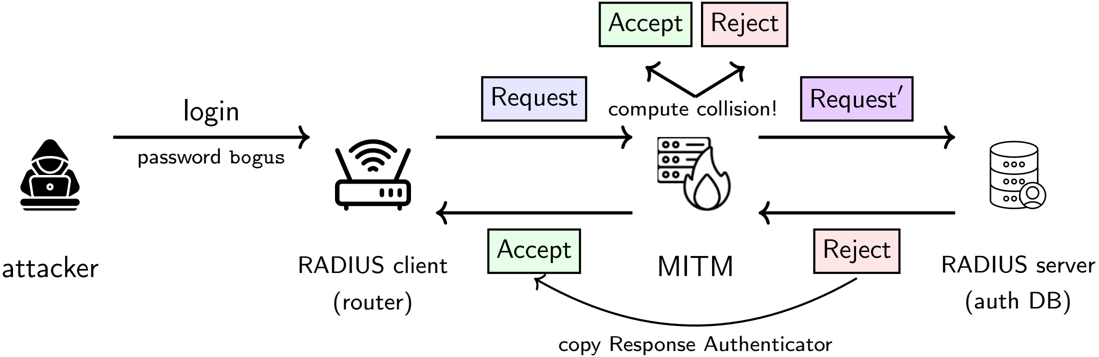


icons from [5]

Blast-RADIUS: Attack Overview

Goal: Forge Access-Accept without knowing shared secret.

Blast-RADIUS attack: Create MD5 collision s.t. Access-Accept and Access-Reject produce same Response Authenticator: MD5(Access-Accept) = MD5(Access-Reject).

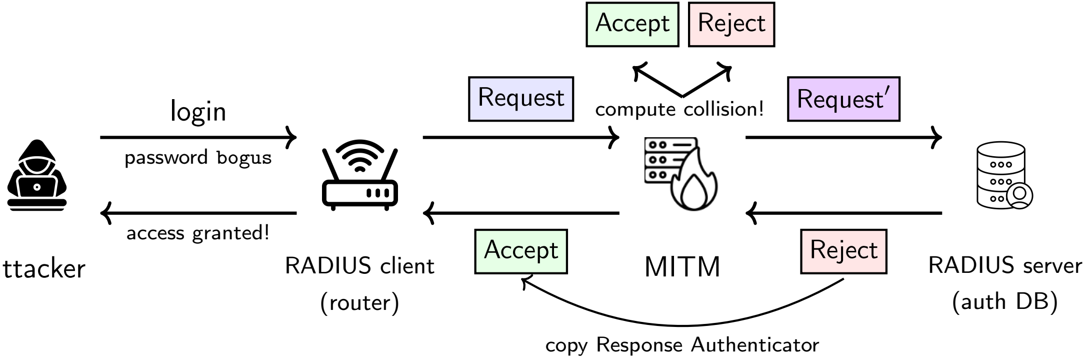


icons from [5]

Blast-RADIUS: Attack Overview

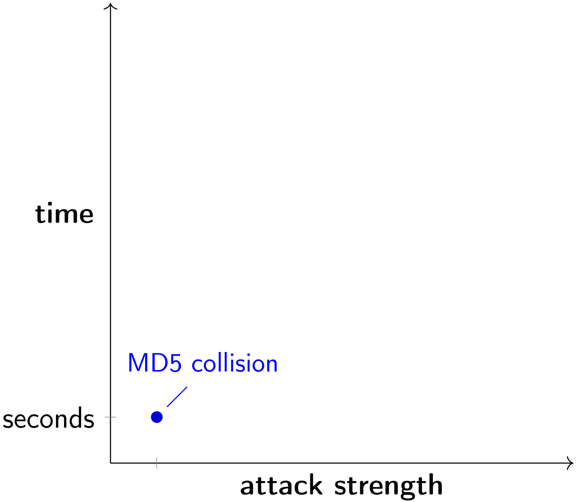
Goal: Forge Access-Accept without knowing shared secret.

Blast-RADIUS attack: Create MD5 collision s.t. Access-Accept and Access-Reject produce same Response Authenticator: MD5(Access-Accept) = MD5(Access-Reject).



icons from [5]

Recap of MD5 Collisions

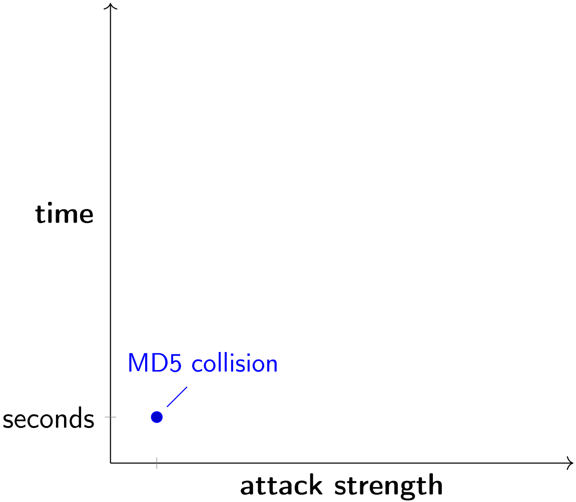


2004: MD5 collision [14]

Produce unstructured strings G_1, G_2 such that

$$\mathrm{MD5}(G_1) = \mathrm{MD5}(G_2).$$

Recap of MD5 Collisions



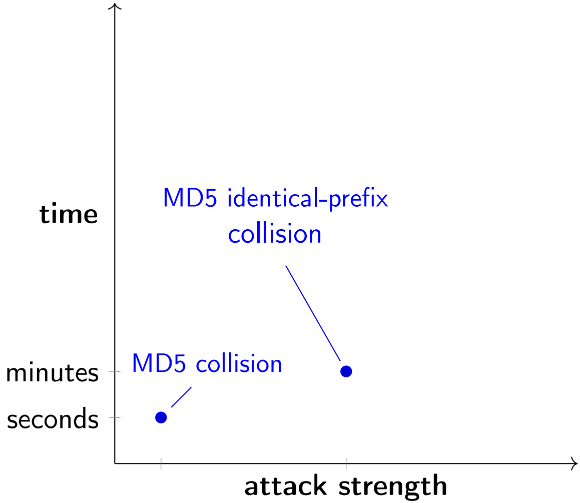
2004: MD5 collision [14]

Produce unstructured strings G_1, G_2 such that

$$\mathrm{MD5}(G_1) = \mathrm{MD5}(G_2).$$

MD5 IS NOT COLLISION-RESISTANT
BUT IT'S STILL OK IN PRACTICE, RIGHT?
imgflip.com

Recap of MD5 Collisions

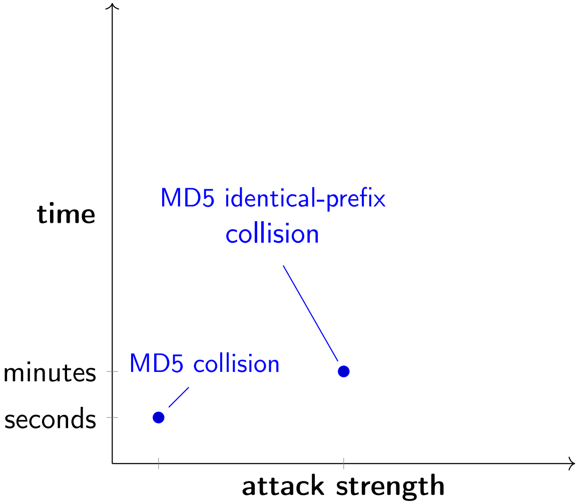


2004: Identical-prefix collision [14]

Given prefix P, produce G_1, G_2 such that

$$\mathrm{MD5}(P||G_1) = \mathrm{MD5}(P||G_2).$$

Recap of MD5 Collisions



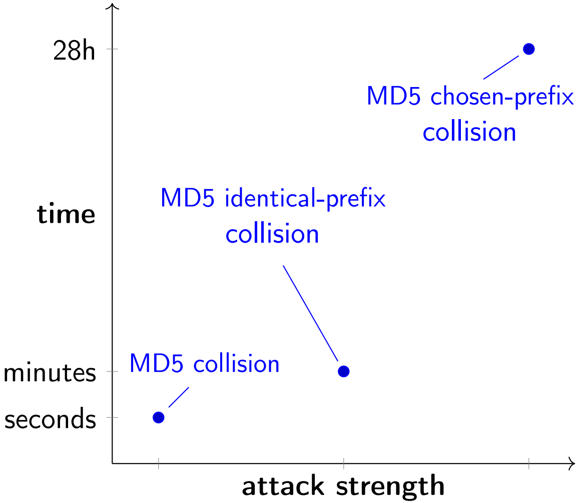
2004: Identical-prefix collision [14]

Given prefix P, produce G_1, G_2 such that

$$\mathrm{MD5}(P||G_1) = \mathrm{MD5}(P||G_2).$$

famous non-MD5 example of an identical-prefix collision [10]

Recap of MD5 Collisions

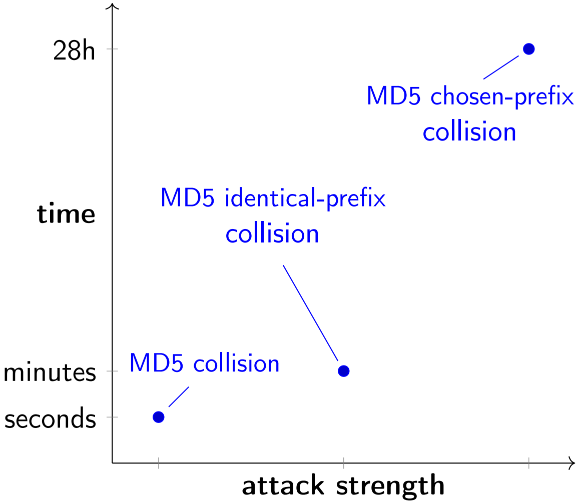


2007: MD5 chosen-prefix collision [11]

Given prefixes P_1, P_2, produce G_1, G_2 such that

$$\mathrm{MD5}(P_1 || G_1) = \mathrm{MD5}(P_2 || G_2).$$

Recap of MD5 Collisions



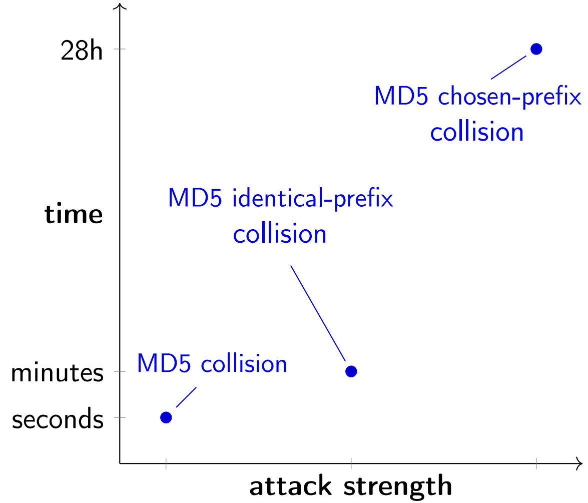
2007: MD5 chosen-prefix collision [11]

Given prefixes P_1, P_2, produce G_1, G_2 such that

$$\mathrm{MD5}(P_1 || G_1) = \mathrm{MD5}(P_2 || G_2).$$

215 PS3 for Rogue TLS CA cert [12]

Recap of MD5 Collisions



2007: MD5 chosen-prefix collision [11]

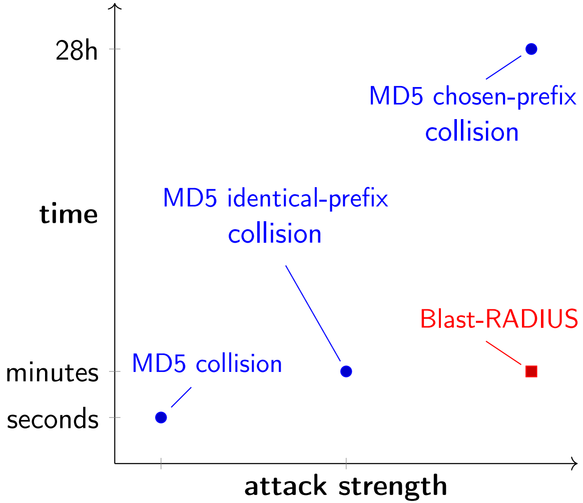
Given prefixes P_1, P_2, produce G_1, G_2 such that

$$\mathrm{MD5}(P_1||G_1) = \mathrm{MD5}(P_2||G_2).$$

Due to Merkle-Damgård structure of MD5, can append identical suffix S:

$$\mathrm{MD5}(P_1||G_1||S) = \mathrm{MD5}(P_2||G_2||S).$$

Recap of MD5 Collisions



2024: Blast-RADIUS

A chosen-prefix collision:*
Given prefixes P_1, P_2, produce G_1, G_2 such that (for any suffix S)

$$\mathrm{MD5}(P_1 || G_1 || S) = \mathrm{MD5}(P_2 || G_2 || S).$$

*But faster and with some additional conditions!

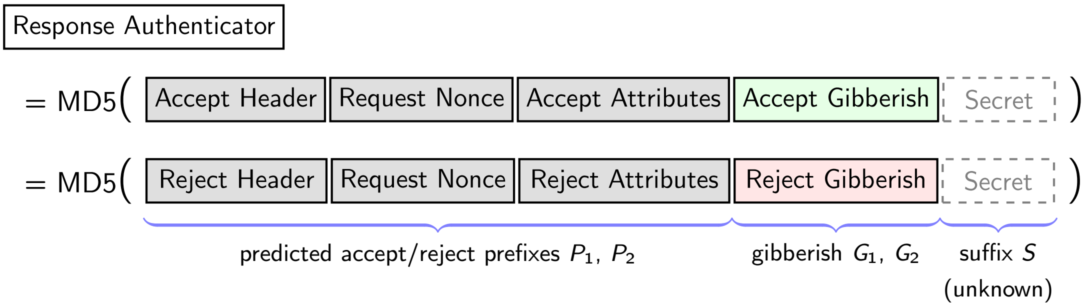
Blast-RADIUS: Turning Access-Reject into Access-Accept

Attack: MD5 collision to forge Access-Accept with same Response Authenticator as Access-Reject (without knowledge of the shared secret).

Blast-RADIUS: Turning Access-Reject into Access-Accept

Attack: MD5 collision to forge Access-Accept with same Response Authenticator as Access-Reject (without knowledge of the shared secret).

MD5 chosen-prefix collision $\mathrm{MD5}(P_1||G_1||S) = \mathrm{MD5}(P_2||G_2||S)$ applied to RADIUS:



Response Authenticator

= MD5(Accept Header | Request Nonce | Accept Attributes | Accept Gibberish | Secret)

= MD5(Reject Header | Request Nonce | Reject Attributes | Reject Gibberish | Secret)

predicted accept/reject prefixes P_1, P_2

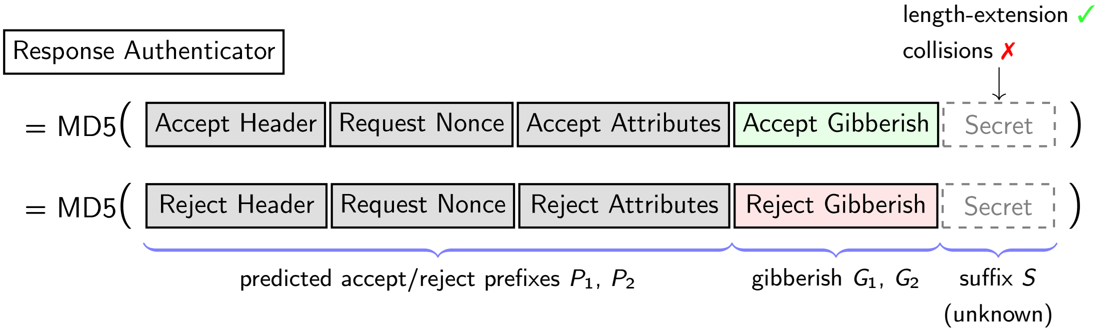
gibberish G_1, G_2

suffix S (unknown)

Blast-RADIUS: Turning Access-Reject into Access-Accept

Attack: MD5 collision to forge Access-Accept with same Response Authenticator as Access-Reject (without knowledge of the shared secret).

MD5 chosen-prefix collision $\mathrm{MD5}(P_1||G_1||S) = \mathrm{MD5}(P_2||G_2||S)$ applied to RADIUS:



length-extension ✓

collisions ✗

Response Authenticator

= MD5(Accept Header | Request Nonce | Accept Attributes | Accept Gibberish | Secret)

= MD5(Reject Header | Request Nonce | Reject Attributes | Reject Gibberish | Secret)

predicted accept/reject prefixes P_1, P_2

gibberish G_1, G_2

suffix S (unknown)

Easy, all done?

"While MD5 has been broken, it is a testament to the design of RADIUS that there have been (as yet) no attacks on RADIUS Authenticator signatures which are stronger than brute-force."

(*"Deprecating Insecure Practices in RADIUS" IETF draft, 2023*)

Challenge 1: Inject MD5 Reject Gibberish In Protocol

Problem: Server must include $\boxed{\text{Reject Gibberish}}$ in Response Authenticator computation for Access-Reject.

$$\text{MD5}\Big(\ \boxed{\text{Reject Header}}\boxed{\text{Request Nonce}}\boxed{\text{Reject Gibberish}}\boxed{\text{Shared Secret}}\ \Big)$$

Challenge 1: Inject MD5 Reject Gibberish In Protocol

Problem: Server must include $\boxed{\text{Reject Gibberish}}$ in Response Authenticator computation for Access-Reject.

$$\mathrm{MD5}\left(\boxed{\text{Reject Header}}\boxed{\text{Request Nonce}}\boxed{\text{Reject Gibberish}}\boxed{\text{Shared Secret}}\right)$$

Solution: The Proxy-State attribute.

> *This Attribute is available to be sent by a proxy server to another server when forwarding an Access-Request and **MUST be returned unmodified** in the Access-Accept, Access-Reject or Access-Challenge.*
>
> *(RFC 2058, emphasis added)*

Challenge 1: Inject MD5 Reject Gibberish In Protocol

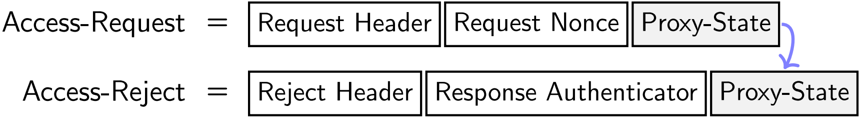
Problem: Server must include Reject Gibberish in Response Authenticator computation for Access-Reject.

MD5(Reject Header | Request Nonce | Reject Gibberish | Shared Secret)

Solution: The Proxy-State attribute.

This Attribute is available to be sent by a proxy server to another server when forwarding an Access-Request and ***MUST be returned unmodified*** *in the Access-Accept, Access-Reject or Access-Challenge.*

(RFC 2058, emphasis added)



Access-Request = Request Header | Request Nonce | Proxy-State

Access-Reject = Reject Header | Response Authenticator | Proxy-State

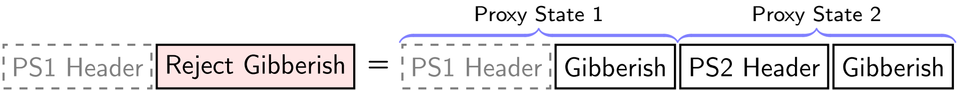
Challenge 2: Fitting Gibberish Into Protocol Attributes

Problem: Hiding Reject Gibberish in single Proxy-State attribute is too slow.

Challenge 2: Fitting Gibberish Into Protocol Attributes

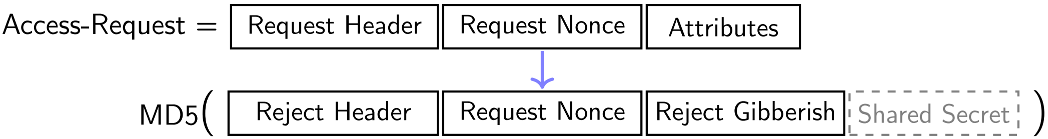
Problem: Hiding Reject Gibberish in single Proxy-State attribute is too slow.

Solution: Spread longer gibberish across multiple Proxy-State attributes by modifying collision algorithm to embed Proxy-State header.



(PS1 Header is part of the MD5 prefix not the gibberish.)

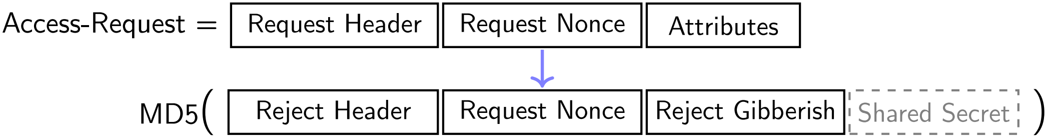
Challenge 3: Fast Collision Computation



Access-Request = Request Header | Request Nonce | Attributes

MD5(Reject Header | Request Nonce | Reject Gibberish | Shared Secret)

Challenge 3: Fast Collision Computation



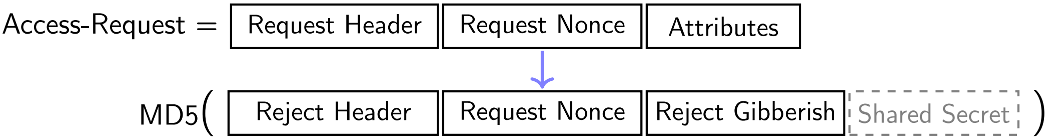
Access-Request = Request Header | Request Nonce | Attributes

MD5(Reject Header | Request Nonce | Reject Gibberish | Shared Secret)

Problem: Computing MD5 prefixes requires $\boxed{\text{Request Nonce}}$.

$\Longrightarrow$ Must compute collision before RADIUS client times out,

Challenge 3: Fast Collision Computation

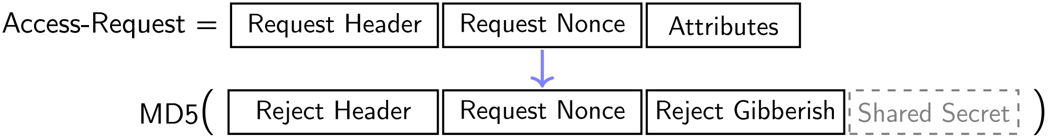


Problem: Computing MD5 prefixes requires $\boxed{\text{Request Nonce}}$.

$\implies$ Must compute collision before RADIUS client times out,

Solution: Reduce collision time from days to $\leq$ 5m (on 47 servers) with algorithmic improvements and parallelization.

Challenge 3: Fast Collision Computation



Access-Request = Request Header | Request Nonce | Attributes

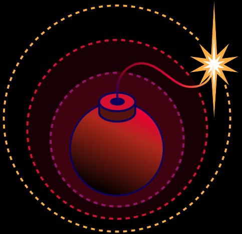
MD5(Reject Header | Request Nonce | Reject Gibberish | Shared Secret)

Problem: Computing MD5 prefixes requires $\boxed{\text{Request Nonce}}$.

$\Longrightarrow$ Must compute collision before RADIUS client times out,

Solution: Reduce collision time from days to $\leq$ 5m (on 47 servers) with algorithmic improvements and parallelization.

Can you go faster? Yes, attack parallelizes well, hardware implementation.



IMPACT & MITIGATION

Impact Summary

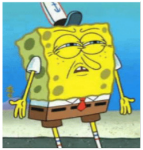
Affected authentication modes:

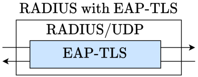
- PAP, CHAP, MS-CHAP are vulnerable.

Impact Summary

Affected authentication modes:

- PAP, CHAP, MS-CHAP are vulnerable.
- EAP not vulnerable due to HMAC-MD5 attribute.

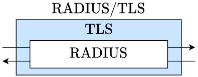




RADIUS with EAP-TLS

RADIUS/UDP

EAP-TLS



RADIUS/TLS

TLS

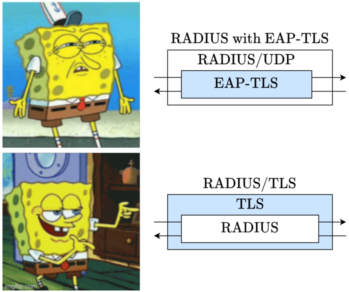
RADIUS

Impact Summary

Affected authentication modes:

- PAP, CHAP, MS-CHAP are vulnerable.
- EAP not vulnerable due to HMAC-MD5 attribute.

Affected deployments: Requires MITM network access



RADIUS with EAP-TLS

RADIUS/UDP

EAP-TLS

RADIUS/TLS

TLS

RADIUS

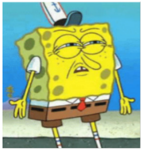
Impact Summary

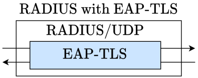
Affected authentication modes:

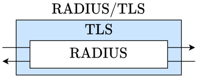
- PAP, CHAP, MS-CHAP are vulnerable.
- EAP not vulnerable due to HMAC-MD5 attribute.

Affected deployments: Requires MITM network access

- RADIUS/UDP traffic over open Internet is vulnerable.







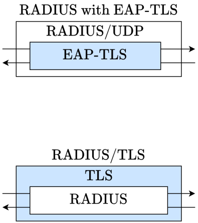
Impact Summary

Affected authentication modes:

- PAP, CHAP, MS-CHAP are vulnerable.
- EAP not vulnerable due to HMAC-MD5 attribute.

Affected deployments: Requires MITM network access

- RADIUS/UDP traffic over open Internet is vulnerable.
  - incl. many non-cable ISPs and major cloud providers.


RADIUS with EAP-TLS

RADIUS/UDP

EAP-TLS

RADIUS/TLS

TLS

RADIUS

Impact Summary

Affected authentication modes:

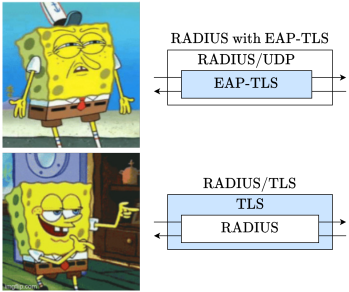
- PAP, CHAP, MS-CHAP are vulnerable.
- EAP not vulnerable due to HMAC-MD5 attribute.

Affected deployments: Requires MITM network access

- RADIUS/UDP traffic over open Internet is vulnerable.
  - incl. many non-cable ISPs and major cloud providers.

"Lots and lots and lots of people [are] sending [RADIUS/]UDP over the public Internet"

(Alan DeKok, FreeRADIUS)



Impact Summary

Affected authentication modes:

- PAP, CHAP, MS-CHAP are vulnerable.
- EAP not vulnerable due to HMAC-MD5 attribute.

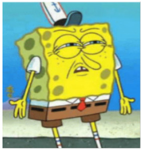
Affected deployments: Requires MITM network access

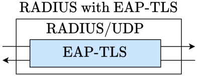
- RADIUS/UDP traffic over open Internet is vulnerable.
  - incl. many non-cable ISPs and major cloud providers.

  "Lots and lots and lots of people [are] sending [RADIUS/]UDP over the public Internet"

  (Alan DeKok, FreeRADIUS)

- RADIUS/UDP traffic over VLAN/IPSEC: useful for lateral movement.

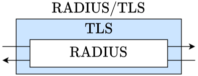




RADIUS with EAP-TLS

RADIUS/UDP

EAP-TLS



RADIUS/TLS

TLS

RADIUS

Mitigations

Massive disclosure with 90+ vendors.

Mitigations

Massive disclosure with 90+ vendors.

Long-term:

- Encapsulate all RADIUS traffic in (D)TLS tunnel.

Mitigations

Massive disclosure with 90+ vendors.

Long-term:

- Encapsulate all RADIUS traffic in (D)TLS tunnel.
- Current IETF draft is being standardized [9].

Mitigations

Massive disclosure with 90+ vendors.

Long-term:

- Encapsulate all RADIUS traffic in (D)TLS tunnel.
- Current IETF draft is being standardized [9].

Challenges: widely used, need backwards compatibility.

Some power plants use RADIUS [13].

Mitigations

Massive disclosure with 90+ vendors.

Long-term:

- Encapsulate all RADIUS traffic in (D)TLS tunnel.
- Current IETF draft is being standardized [9].

Challenges: widely used, need backwards compatibility.

Short-term:

- Message-Authenticator attribute uses HMAC-MD5 not vulnerable to MD5 collisions.

Some power plants use RADIUS [13].

Mitigations

Massive disclosure with 90+ vendors.

Long-term:

- Encapsulate all RADIUS traffic in (D)TLS tunnel.
- Current IETF draft is being standardized [9].

Challenges: widely used, need backwards compatibility.

Short-term:

- Message-Authenticator attribute uses HMAC-MD5 not vulnerable to MD5 collisions.
- All requests and responses should include and verify Message-Authenticator.

Some power plants use RADIUS [13].

Mitigations Trouble

Many equipment vendors have upgraded [2], but some challenges remain:

Mitigations Trouble

Many equipment vendors have upgraded [2], but some challenges remain:

- Juniper vs. Cisco: incompatible Message-Authenticator placement.

Mitigations Trouble

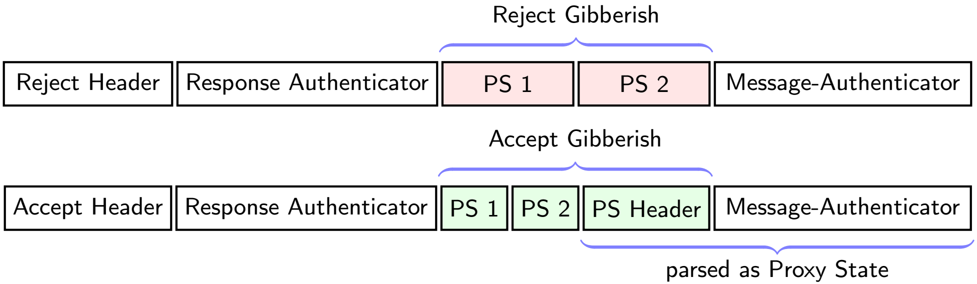
Many equipment vendors have upgraded [2], but some challenges remain:

- Juniper vs. Cisco: incompatible Message-Authenticator placement.
- Correct behavior: put as first attribute for sending, mandate presence for receiving.

Mitigations Trouble

Many equipment vendors have upgraded [2], but some challenges remain:

- Juniper vs. Cisco: incompatible Message-Authenticator placement.
- Correct behavior: put as first attribute for sending, mandate presence for receiving.
- Incorrect placement may be vulnerable to **Message-Authenticator hiding attack**:



Blast-RADIUS Attack

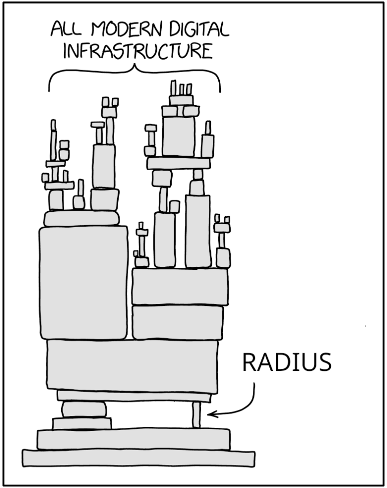
Attack summary: MD5 collision attack on RADIUS authentication by MITM adversary.

`https://blastradius.fail`

RADIUS/UDP Considered Harmful

Sharon Goldberg, Miro Haller, Nadia Heninger, Mike Milano, Dan Shumow, Marc Stevens, and Adam Suhl.

USENIX Security, August 2024.

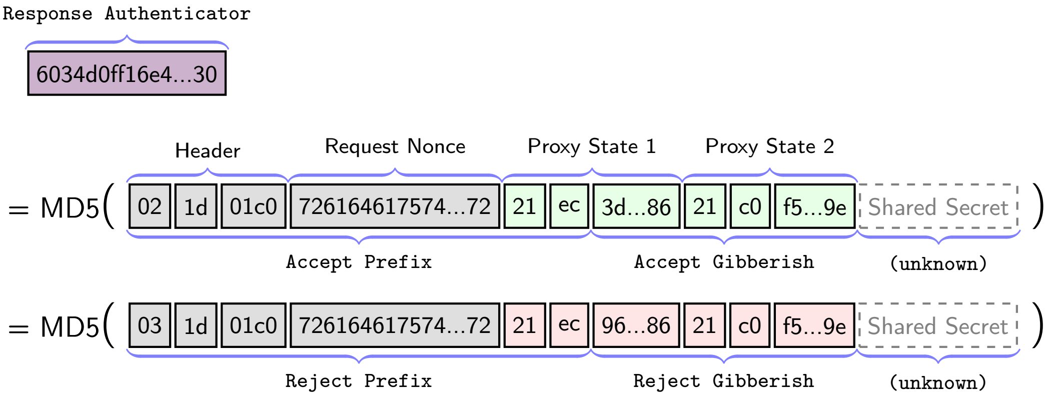


XKCD modified from [7]

BONUS MATERIAL

Blast-RADIUS: Example

As concrete example, putting everything together, we get the following collision.



Response Authenticator

6034d0ff16e4…30

Header

Request Nonce

Proxy State 1

Proxy State 2

= MD5(02 | 1d | 01c0 | 726164617574…72 | 21 | ec | 3d…86 | 21 | c0 | f5…9e | Shared Secret)

Accept Prefix

Accept Gibberish

(unknown)

= MD5(03 | 1d | 01c0 | 726164617574…72 | 21 | ec | 96…86 | 21 | c0 | f5…9e | Shared Secret)

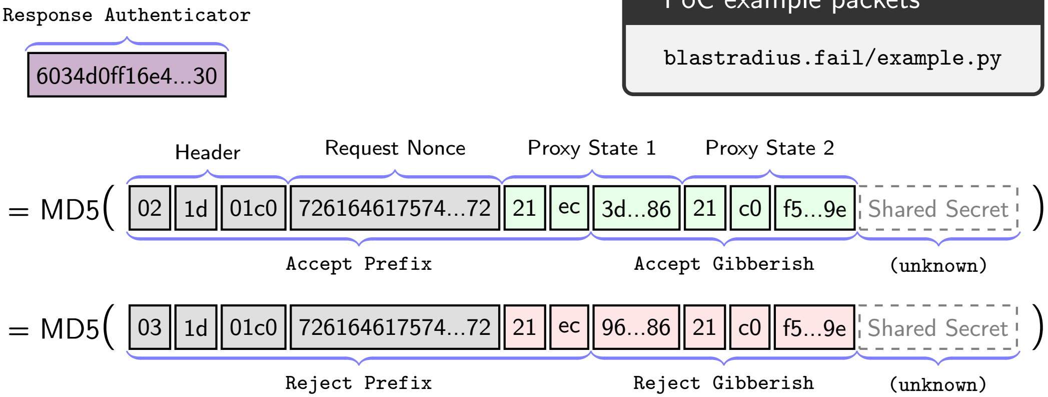
Reject Prefix

Reject Gibberish

(unknown)

Blast-RADIUS: Example

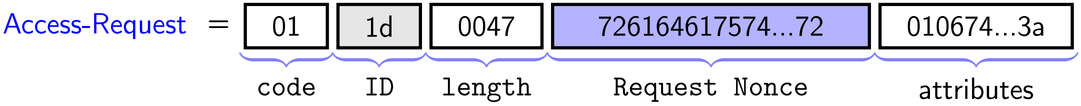
As concrete example, putting everything together, we get the following collision.



PoC example packets

`blastradius.fail/example.py`

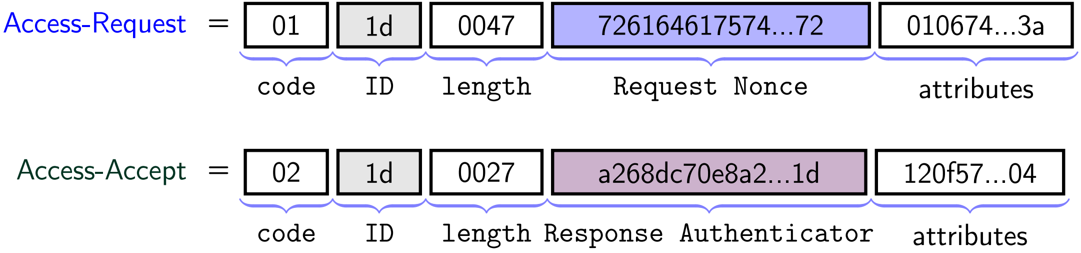
End-to-End Example Attack (1/4)



Access-Request = 01 1d 0047 726164617574...72 010674...3a

code ID length Request Nonce attributes

End-to-End Example Attack (1/4)



Access-Request = 01 | 1d | 0047 | 726164617574...72 | 010674...3a

code | ID | length | Request Nonce | attributes

Access-Accept = 02 | 1d | 0027 | a268dc70e8a2...1d | 120f57...04

code | ID | length | Response Authenticator | attributes

End-to-End Example Attack (1/4)

Access-Request	=	01	1d	0047	726164617574...72	010674...3a
		code	ID	length	Request Nonce	attributes
Access-Accept	=	02	1d	0027	a268dc70e8a2...1d	120f57...04
		code	ID	length	Response Authenticator	attributes
Access-Reject	=	03	1d	0024	357bf27e8c0a...e5	121041...2e
		code	ID	length	Response Authenticator	attributes

End-to-End Example Attack (2/4)

PoC example packets

`blastradius.fail/example.py`

End-to-End Example Attack (2/4)

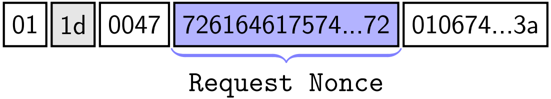
1. Attacker triggers Access-Request.

PoC example packets

`blastradius.fail/example.py`

End-to-End Example Attack (2/4)

1. Attacker triggers Access-Request.
2. MITM attacker observes Access-Request.



01	1d	0047	726164617574...72	010674...3a

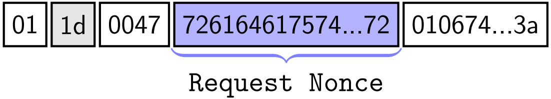
Request Nonce

PoC example packets

`blastradius.fail/example.py`

End-to-End Example Attack (2/4)

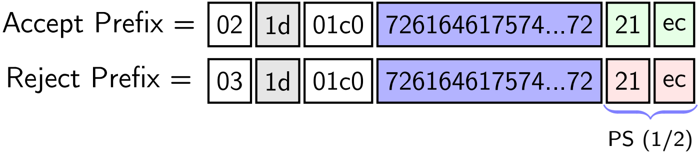
1. Attacker triggers Access-Request.
2. MITM attacker observes Access-Request.



PoC example packets

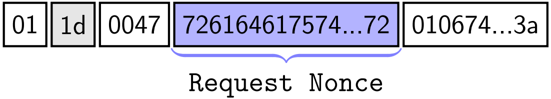
`blastradius.fail/example.py`

3. MITM attacker predicts the following prefixes



End-to-End Example Attack (2/4)

1. Attacker triggers Access-Request.
2. MITM attacker observes Access-Request.



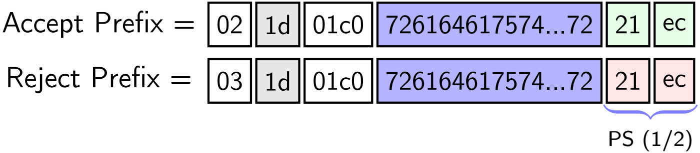
01 | 1d | 0047 | 726164617574...72 | 010674...3a

Request Nonce

PoC example packets

`blastradius.fail/example.py`

3. MITM attacker predicts the following prefixes

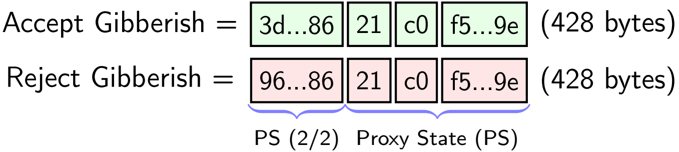


Accept Prefix = 02 | 1d | 01c0 | 726164617574...72 | 21 | ec

Reject Prefix = 03 | 1d | 01c0 | 726164617574...72 | 21 | ec

PS (1/2)

to compute the MD5 chosen-prefix collision gibberish.



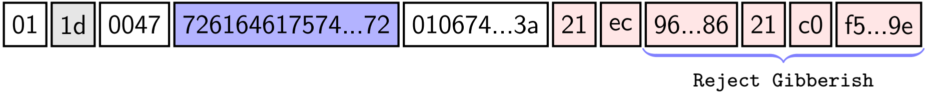
Accept Gibberish = 3d...86 | 21 | c0 | f5...9e (428 bytes)

Reject Gibberish = 96...86 | 21 | c0 | f5...9e (428 bytes)

PS (2/2) Proxy State (PS)

End-to-End Example Attack (3/4)

4. MITM sends Access-Request with appended `Reject Gibberish` to server.

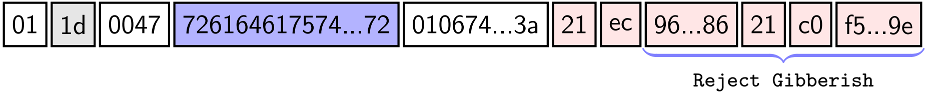


01 | 1d | 0047 | 726164617574...72 | 010674...3a | 21 | ec | 96...86 | 21 | c0 | f5...9e

`Reject Gibberish`

End-to-End Example Attack (3/4)

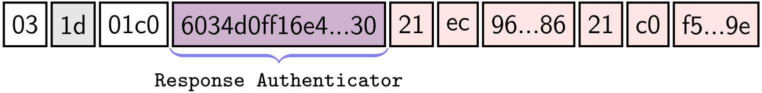
4. MITM sends Access-Request with appended `Reject Gibberish` to server.



01	1d	0047	726164617574...72	010674...3a	21	ec	96...86	21	c0	f5...9e

`Reject Gibberish`

5. MITM intercepts Access-Reject, learning the Response Authenticator.

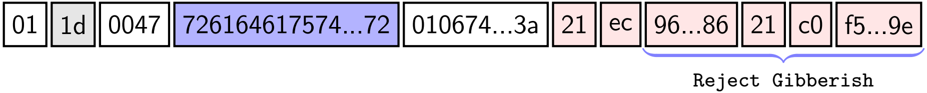


03	1d	01c0	6034d0ff16e4...30	21	ec	96...86	21	c0	f5...9e

`Response Authenticator`

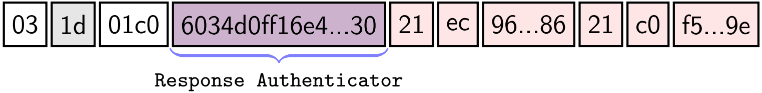
End-to-End Example Attack (3/4)

4. MITM sends Access-Request with appended `Reject Gibberish` to server.



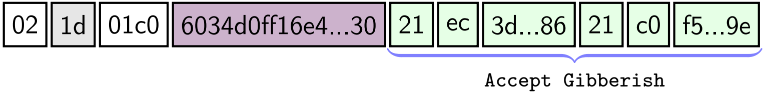
01	1d	0047	726164617574...72	010674...3a	21	ec	96...86	21	c0	f5...9e

5. MITM intercepts Access-Reject, learning the Response Authenticator.



03	1d	01c0	6034d0ff16e4...30	21	ec	96...86	21	c0	f5...9e

6. MITM puts Response Authenticator in Access-Accept packet with appended `Accept Gibberish`.



02	1d	01c0	6034d0ff16e4...30	21	ec	3d...86	21	c0	f5...9e

End-to-End Example Attack (4/4)

7. Access-Accept and Access-Reject produce the same Response Authenticator, and hence pass the RADIUS client authentication check.

End-to-End Example Attack (4/4)

7. Access-Accept and Access-Reject produce the same Response Authenticator, and hence pass the RADIUS client authentication check.

$$\overbrace{\boxed{\text{6034d0ff16e4...30}}}^{\texttt{Response Authenticator}}$$

$$= \mathrm{MD5}\Big(\underbrace{\boxed{02}\,\boxed{1d}\,\boxed{01c0}\,\boxed{726164617574...72}\,\boxed{21}\,\boxed{ec}}_{\texttt{Accept Prefix}}\,\underbrace{\boxed{3d...86}\,\boxed{21}\,\boxed{c0}\,\boxed{f5...9e}}_{\texttt{Accept Gibberish}}\,\underbrace{\boxed{\text{Shared Secret}}}_{\texttt{(unknown)}}\Big)$$

$$= \mathrm{MD5}\Big(\underbrace{\boxed{03}\,\boxed{1d}\,\boxed{01c0}\,\boxed{726164617574...72}\,\boxed{21}\,\boxed{ec}}_{\texttt{Reject Prefix}}\,\underbrace{\boxed{96...86}\,\boxed{21}\,\boxed{c0}\,\boxed{f5...9e}}_{\texttt{Reject Gibberish}}\,\underbrace{\boxed{\text{Shared Secret}}}_{\texttt{(unknown)}}\Big)$$

Successful PoCs*

Blast-RADIUS allows attacker to authenticate:

*With longer timeouts than used in practice.

Successful PoCs*

Blast-RADIUS allows attacker to authenticate:

- **FreeRADIUS 3.2.3**: “most widely used RADIUS server in the world” [8]

*With longer timeouts than used in practice.

Successful PoCs*

Blast-RADIUS allows attacker to authenticate:

- **FreeRADIUS 3.2.3**: “most widely used RADIUS server in the world” [8]
- **Okta**: RADIUS in PAP mode for MFA.

*With longer timeouts than used in practice.

Successful PoCs*

Blast-RADIUS allows attacker to authenticate:

- **FreeRADIUS 3.2.3**: "most widely used RADIUS server in the world" [8]
- **Okta**: RADIUS in PAP mode for MFA.
- **Cisco ASA 5505 firewall** using RADIUS to authenticate users for access to serial console, VPN, Telnet, FTP, or HTTPS.

PoC with Cisco ASA 5505 firewall tunneling UDP via TCP to our cluster.

*With longer timeouts than used in practice.

Successful PoCs*

Blast-RADIUS allows attacker to authenticate:

- **FreeRADIUS 3.2.3**: “most widely used RADIUS server in the world” [8]
- **Okta**: RADIUS in PAP mode for MFA.
- **Cisco ASA 5505 firewall** using RADIUS to authenticate users for access to serial console, VPN, Telnet, FTP, or HTTPS.
- **PAM**: RADIUS authentication for SSH, sudo.

PoC with Cisco ASA 5505 firewall tunneling UDP via TCP to our cluster.

*With longer timeouts than used in practice.

Successful PoCs*

Blast-RADIUS allows attacker to authenticate:

- **FreeRADIUS 3.2.3**: “most widely used RADIUS server in the world” [8]
- **Okta**: RADIUS in PAP mode for MFA.
- **Cisco ASA 5505 firewall** using RADIUS to authenticate users for access to serial console, VPN, Telnet, FTP, or HTTPS.
- **PAM**: RADIUS authentication for SSH, sudo.

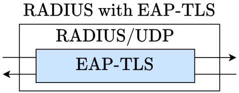
⟹ Confirms no Message-Authenticator used, Proxy-State accepted in Access-Accept.

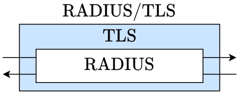
PoC with Cisco ASA 5505 firewall tunneling UDP via TCP to our cluster.

*With longer timeouts than used in practice.

EAP: It's Complicated.

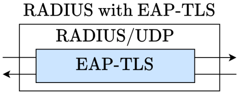
- TLS in EAP-TLS does not protect RADIUS packets.
- Not to be confused with RADIUS/TLS, which properly nests RADIUS inside TLS.





EAP: It's Complicated.

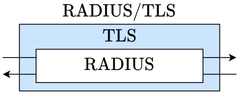
- TLS in EAP-TLS does not protect RADIUS packets.
- Not to be confused with RADIUS/TLS, which properly nests RADIUS inside TLS.
- RFC 3579 requires that EAP-Message has Message-Authenticator attribute [1].



RADIUS with EAP-TLS

RADIUS/UDP

EAP-TLS



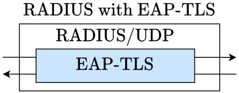
RADIUS/TLS

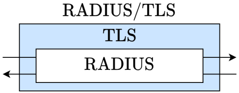
TLS

RADIUS

EAP: It's Complicated.

- TLS in EAP-TLS does not protect RADIUS packets.
- Not to be confused with RADIUS/TLS, which properly nests RADIUS inside TLS.
- RFC 3579 requires that EAP-Message has Message-Authenticator attribute [1].
- Unclear client behavior for Access-Accept without EAP-Message.

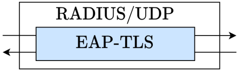




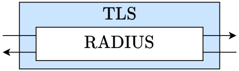
EAP: It’s Complicated.

- TLS in EAP-TLS does not protect RADIUS packets.
- Not to be confused with RADIUS/TLS, which properly nests RADIUS inside TLS.
- RFC 3579 requires that EAP-Message has Message-Authenticator attribute [1].
- Unclear client behavior for Access-Accept without EAP-Message.
- In eduroam and 802.1X, key is negotiated inside EAP session $\implies$ would require further attacks.

RADIUS with EAP-TLS



RADIUS/TLS

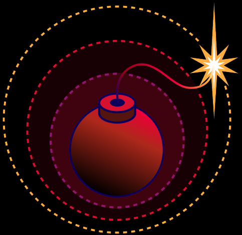


Attack Extensions

- Adversary can add arbitrary attributes in prefix for Access-Accept.

$$\text{AcceptPrefix} = \boxed{02}\,\boxed{1d}\,\boxed{01c0}\,\boxed{726164617574...72}\,\underbrace{\boxed{1a0b000007db1d04}}_{\substack{\text{Attribute:} \\ \texttt{Exec-Privilege 04}}}\,\boxed{21}\,\boxed{ec}$$

- Proxy-State attributes are *not* the only way to inject the `RejectGibberish`.
  - Any reflected user input could work, e.g. User-Name or Vendor-Specific attributes.
    - In Access-Request:
      `User-Name:  0PZjN-_ayr83S-nc6q...Mt85`
    - In Access-Reject:
      `Reply-Message:  Login for 0PZjN-_ayr83S-nc6q...Mt85 failed!`
  - The client does not need to support or parse these attributes.



REFERENCES

References I

[1] Pat R. Calhoun and Dr. Bernard D. Aboba. *RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)*. RFC 3579. Sept. 2003. DOI: 10.17487/RFC3579. URL: https://www.rfc-editor.org/info/rfc3579.

[2] Alan DeKok. *Personal Communication*.

[3] Alan DeKok. *[radext] BlastRADIUS problems*. https://mailarchive.ietf.org/arch/msg/radext/7c9-35Xh6I-oCTJ4_xydvpczGPU/. Sept. 2024.

[4] Alan DeKok. *RADIUS and MD5 Collision Attacks*. https://networkradius.com/assets/pdf/radius_and_md5_collisions.pdf. 2024.

[5] *Icons from https://www.flaticon.com*. visited on Dec 4, 2024.

References II

[6] *[ISE Message-Authenticator Attribute order*. https://community.cisco.com/t5/network-access-control/ise-message-authenticator-attribute-order/td-p/5205188. Oct. 2024.

[7] Randall Munroe. *Dependency (Comic #2347)*. https://www.xkcd.com/2347/. Aug. 2020.

[8] The FreeRADIUS Server Project and Contributors. *FreeRADIUS*. https://freeradius.org/.

[9] Jan-Frederik Rieckers and Stefan Winter. *(Datagram) Transport Layer Security ((D)TLS Encryption for RADIUS*. Internet-Draft draft-ietf-radext-radiusdtls-bis-02. Work in Progress. Internet Engineering Task Force, July 2024. 38 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-radext-radiusdtls-bis/02/.

[10] *SHAttered*. https://shattered.io/. Accessed March 20, 2025.

References III

[11] Marc Stevens, Arjen K. Lenstra, and Benne de Weger. "Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities". In: *EUROCRYPT*. Vol. 4515. Lecture Notes in Computer Science. Springer, 2007, pp. 1–22.

[12] Marc Stevens et al. "Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate". In: *CRYPTO*. Vol. 5677. Lecture Notes in Computer Science. Springer, 2009, pp. 55–69.

[13] Henrik Thejl, Nagaraja K S, and Karl-Georg Aspacher. "A method for user management and a power plant control system thereof for a power plant system". Pat. 2765466. Siemens Gamesa Renewable Energy A/S. Jan. 24, 2014. URL: https://data.epo.org/publication-server/rest/v1.0/publication-dates/20190904/patents/EP2765466NWB1/document.pdf.

References IV

[14] Xiaoyun Wang and Hongbo Yu. “How to Break MD5 and Other Hash Functions”. In: *EUROCRYPT*. Vol. 3494. Lecture Notes in Computer Science. Springer, 2005, pp. 19–35.