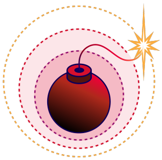
The Blast-RADIUS Attack

Sharon Goldberg[1], **Nadia Heninger**[2], **Miro Haller**[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

July 26, 2024



Attack Summary

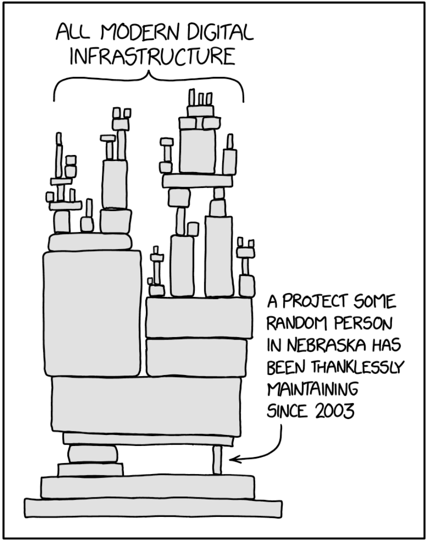
A man-in-the-middle network attacker can forge arbitrary RADIUS responses for non-EAP authentication modes:

- Turning an Access-Reject into an Access-Accept
- Adding arbitrary network access attributes to Access-Accept.

This is a **protocol vulnerability**: RADIUS hard codes weak authentication based on broken MD5 hash function.

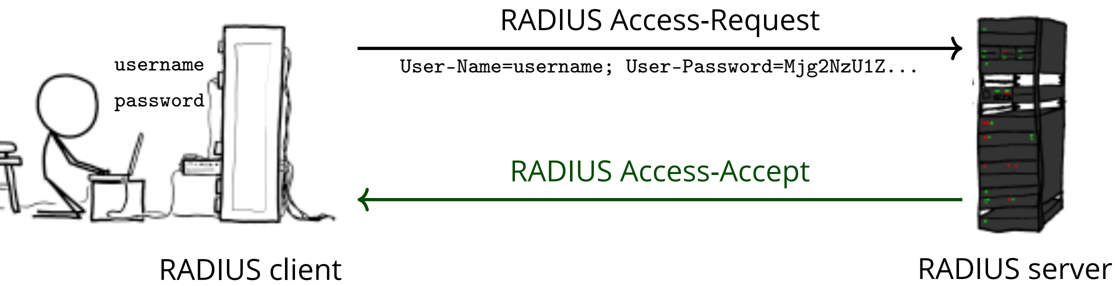
Our paper will appear at USENIX Security 2024 and is available on `https://www.blastradius.fail`.

Obligatory XKCD



- RADIUS is the de facto standard lightweight protocol for authentication, authorization, and accounting (AAA) for networked devices.
- RADIUS is *everywhere*: ISPs (DSL/FTTH), 802.1X, WiFi, mobile roaming, IoT, router admin access...

RADIUS Protocol Reminder



- RADIUS requests and responses are often sent over UDP.
- RFC 6614 TLS Encryption for RADIUS (2012) never left experimental status.
- (D)TLS Encryption for RADIUS: `draft-ietf-radext-radiusdtls-bis`

Apologies to XKCD

RADIUS Packet Formats

Access-Request

01	1d	0047	726164617574...72	010674...3a
`code`	`ID`	`length`	`Request Authenticator`	Attributes:

Attributes:

```
User-Name test
Password Mjg2NzU1z
NAS-Identifier = localhost
```

- Access-Request packet contents unauthenticated.
- UDP source IP address is used to identify/validate client.
- Client and server share fixed shared secret.
- Passwords obfuscated using MD5+shared secret.

RADIUS Packet Formats

Access-Accept

02	1d	0027	a268dc70e8a294e9...1d	120f57...04
Code	ID	length	Response Authenticator	Attributes:

`Reply-Message Welcome test!`

`Exec-Privilege 4`

Access-Reject

03	1d	0024	357bf27e8c0a9961...e5	121041...2e
code	id	length	Response Authenticator	Attributes:

`Reply-Message Access denied.`

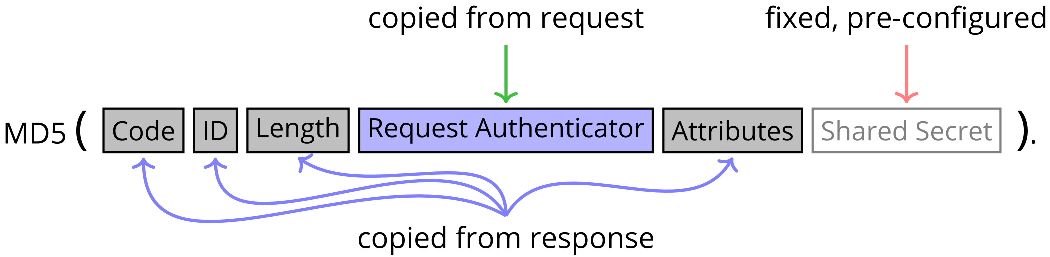
- Response Authenticator: Ad hoc authentication with MD5 hash.

Response Authenticator

Goal: Prevent forgery of packets, e.g., by man-in-the-middle attacker.

The Response Authenticator from packet

is computed as



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

- Our attack allows a MITM attacker to produce a valid Response Authenticator *without* knowing the Shared Secret.

- It creates an MD5 collision such that Access-Accept and Access-Reject produce the same Response Authenticator (very simplifed):

  MD5(Access-Accept) = MD5(Access-Reject).

MD5 Collision Attack History

1993 Known weaknesses in MD5.

2004 First full MD5 collision. Produced unstructured strings G_1, G_2 with

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

$\implies$ Unstructured G_1, G_2 hard to exploit in realistic contexts.

2004 Identical-prefix collision. Given prefix P, produce G_1, G_2 such that

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

$\implies$ Takes seconds on a laptop. Used to create colliding PDFs etc.

MD5 Collision Attack History

2007 Chosen-prefix collision. Given prefixes P_1 and P_2, produce G_1 and G_2 such that

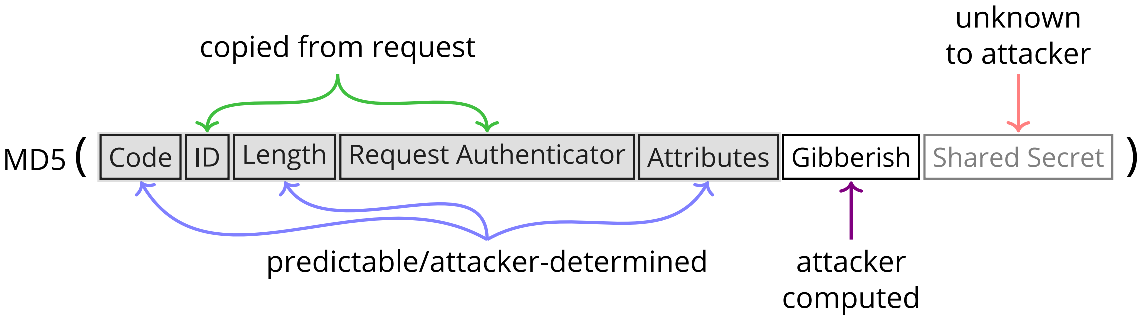
$$\mathrm{MD5}(P_1||G_1) = \mathrm{MD5}(P_2||G_2).$$

Because of MD5 structure, can append any fixed suffix S and still collide:

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

$\implies$ This is what we want to exploit in a real protocol.

A RADIUS Response Authenticator MD5 Collision



1. Given a RADIUS request and two possible responses, an attacker can add an attribute that causes them to have colliding Response Authenticators.
2. E.g., a forged Access-Accept and expected observed Access-Reject.
3. Attacker can copy valid Response Authenticator from observed response to desired forged response.

A RADIUS Response Authenticator MD5 Collision

Hence, the adversary's goal is to compute the following chosen prefix collision:

MD5(AcceptPrefix | AcceptGibberish | Shared Secret)

=

MD5(RejectPrefix | RejectGibberish | Shared Secret)

Problem 1: Collision prefixes `AcceptPrefix` and `RejectPrefix` depend on Request Authenticator and ID from Access Request.

Problem 2: Attacker needs to hide collision gibberish `AcceptGibberish` in an attribute that is echoed back from the server.

Resolving Problem 1: Online Collision Computation

Problem 1: Collision prefixes `AcceptPrefix` and `RejectPrefix` depend on Request Authenticator and ID from Access Request.

Solution: Attacker

1. intercepts request from victim client,
2. learns Request Authenticator,
3. predicts prefixes,
4. and computes MD5 collision before client timeout.

We optimized collision to get time from multiple hours to under 5 minutes for proof-of-concept attack.

We could have decreased it further by implementing on GPU/FPGA but did not think this was a good use of grad student time.

Resolving Problem 2: Reflection Via Proxy-State Attribute

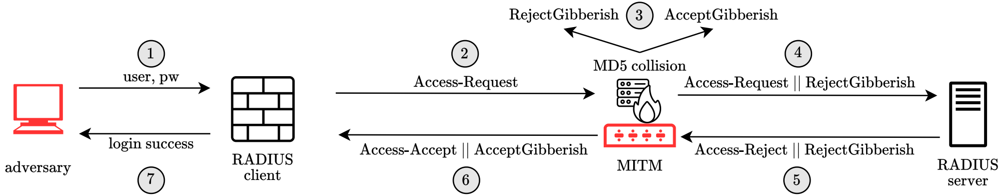
Problem 2: Attacker needs to hide collision gibberish in an attribute that is echoed back from the server.

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)*

$\Longrightarrow$ Attacker intercepts Access-Request and injects malicious Proxy-State attribute to force collision.

Blast-RADIUS Attack Overview



1. Adversary logs into victim NAS with invalid password.
2. Victim NAS makes RADIUS Access-Request to RADIUS server.
3. MITM intercepts request and computes MD5 collision.
4. MITM injects collision gibberish into Proxy-State attribute in request.
5. Victim RADIUS server rejects request.
6. MITM copies Response Authenticator from reject into forged Access-Accept.
7. Victim NAS RADIUS client receives forged accept and permits login.

Blast-RADIUS Attack Example (1/3)

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

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

`Request Authenticator` (726164617574...72)

PoC example packets

`blastradius.fail/example.py`

3. MITM attacker predicts the following prefixes

AcceptPrefix = | 02 | 1d | 01c0 | 726164617574...72 |

RejectPrefix = | 03 | 1d | 01c0 | 726164617574...72 |

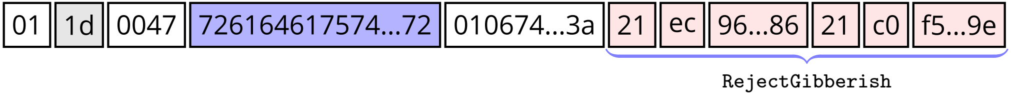
to compute the MD5 chosen-prefix collision gibberish.

AcceptGibberish =	21	ec	3d...86	21	c0	f5...9e	(428 bytes)
RejectGibberish =	21	ec	96...86	21	c0	f5...9e	(428 bytes)

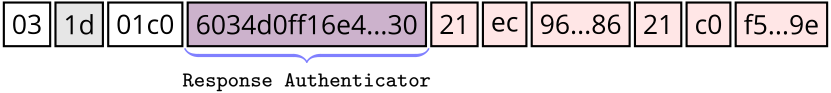
Proxy State (21 ec ...86) — Proxy State (21 c0 f5...9e)

Blast-RADIUS Attack Example (2/3)

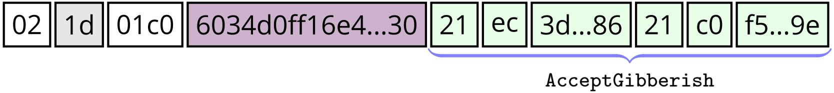
4. MITM sends Access-Request with appended `RejectGibberish` to server.



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

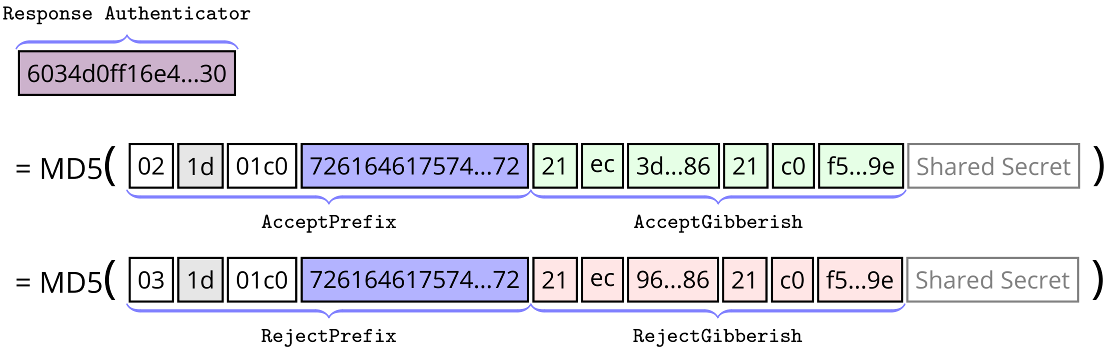


6. MITM puts Response Authenticator in Access-Accept packet with appended `AcceptGibberish`.



Blast-RADIUS Attack Example (3/3)

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



Attack Extensions

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

AcceptPrefix = | 02 | 1d | 01c0 | 726164617574...72 | 1a0b000007db1d04 |

Attribute:

`Exec-Privilege 04`

- Proxy-State attributes are *not* the only way to inject the `RejectGibberish`.
  - Any reflected user input could work, e.g. the 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.

Impact and Mitigation

Impact:

- PAP, CHAP, MS-CHAP are vulnerable
- Modes *requiring* a Message-Authenticator attribute are not vulnerable.
  - E.g., EAP.
  - HMAC-MD5 is not vulnerable to MD5 collision attack.

Threat model: Requires MITM network access

- RADIUS/UDP traffic over open internet is exposed/vulnerable.
- RADIUS/UDP traffic over VLAN or IPSEC requires attacker to have network access to exploit; useful for lateral movement within org.

Short-term mitigation: All requests and responses should include and verify Message-Authenticator attribute: `draft-ietf-radext-deprecating-radius-02`.

Long-term mitigation: All RADIUS traffic should be encapsulated in (D)TLS tunnel: `draft-ietf-radext-radiusdtls-bis-02`.

Discussion

MD5 has been known to have weaknesses for 30 years.

MD5 has been cryptographically broken for 20 years.

The absence of an explicit attack against MD5 in a given protocol likely indicates cryptographers are unaware it is still in use, rather than security.

Shout out to CERT and Alan for coordinating disclosure process.

Blast-RADIUS Takeaways

Attack summary: Man-in-the-middle attack allowing arbitrary forged RADIUS responses for non-EAP authentication methods.

Long-term mitigation: Standardize and move to RADIUS/TLS.

RADIUS/UDP Considered Harmful
Sharon Goldberg, Miro Haller, Nadia Heninger, Mike Milano, Dan Shumow, Marc Stevens, and Adam Suhl.
To appear at USENIX Security, August 2024.

`https://www.blastradius.fail`