#! Anatomy of a Bug

Autopsy of CVSS 8.7

The MongoDB Zero-Day

The Patient

- **Name:** MongoDB Wire Protocol
- **CVE:** CVE-2025-14847
- **Diagnosis:** Pre-auth Memory Disclosure
- **Vector:** Remote Network
- **Severity:** 🔥 8.7

What does 8.7 mean?

- **Auth:** None
- **User Interaction:** None
- **Attack Vector:** Remote
- **Result:** Systemic Confidentiality Loss (Heap Dump)

#1 The Context

The Holiday Exploit

#1.1 The Context

The Christmas Exploit

👉 **Disclosure Date:** December 25, 2025

👉 **Strategic Timing:** Mirrors Log4j (2021) and SolarWinds (2020)

👉 **The Fallacy:** "Holiday Freezes" protect infrastructure.

👉 SOC staffing is **minimal.**

👉 Change freezes **prevent rapid patching.**

👉 **Reality:** Attackers use this "dwell time" to **harvest data before defenders return.**

#1.2 The Context

The Internal Trust Failure

👉 **Database Exposure:** 87,000+ instances **publicly reachable.**

👉 **Cloud Reality:** 42% of cloud environments have **vulnerable instances (Wiz).**

👉 **Supply Chain:** Embedded in appliances like **Ubiquiti UniFi Controllers.**

👉 **Assumption:** "Internal databases don't need strict segmentation."

👉 **Result:** A single compromised web server allows **internal pivoting to the DB via MongoBleed.**

#2 The Architecture

Wire Protocol & Compression

#2.1 The Architecture

The MongoDB Wire Protocol

👉 **Layered over TCP/IP:** The nervous system of the cluster.

👉 **Evolution of Opcodes:**

- 👉 Legacy: **OP_QUERY** (2004)
- 👉 Modern: **OP_MSG** (2013)
- 👉 Generic BSON envelope.
- 👉 Optimization: **OP_COMPRESSED** (2012)

👉 **OP_COMPRESSED Structure:**

- 👉 **originalOpcode:** What's inside.
- 👉 **uncompressedSize:** The fatal field. A hint for allocation.
- 👉 **compressorId:** 1=Snappy, 2=Zlib, 3=Zstd.

#2.2 The Architecture

Compression Negotiation

👉 Compression is optional but negotiated at handshake.

👉 Zlib (ID 2):

👉 High compression ratio.

👉 Default in many package manager distributions.

👉 Computationally expensive, but bandwidth efficient.

👉 The Flaw is specific to Zlib implementation in message_compressor_zlib.cpp. Snappy/Zstd are safe.

#3 The Vulnerability

Anatomy of the Leak

#3.1 The Vulnerability

The Logic Error

👉 Location:
src/mongo/transport/message_compressor_zlib.cpp

👉 The Sequence:

👉 Server reads header: uncompressedSize = 1MB.

👉 Server allocates 1MB from heap (tcmalloc). Memory is dirty.

👉 Server calls zlib::inflate on payload ("A"). Writes 1 byte.

👉 The Bug: The wrapper returns output.length() (Capacity: 1MB) instead of bytes_written (1 Byte).

#3.2 The Vulnerability

The Reflection (Read Oracle)

👉 **Attacker Input:**

👉 Claims: **Payload is 1MB**

👉 Sends: **Compressed letter 'A'**

👉 **Server Action:**

👉 Allocates **1MB.**

👉 Writes **A** at **index 0.**

👉 Leaves **1,048,575** bytes of **uninitialized heap data** untouched.

👉 **Protocol Failure:**

👉 Server treats the **whole buffer as valid.**

👉 Attempts to parse or reflect it.

👉 Sends the **dirty memory back to the attacker.**

#3.3 The Vulnerability

Data at Risk

👉 **Authentication Artifacts:** SCRAM-SHA-256 hashes, plaintext passwords (TLS term).

👉 **BSON Documents:** Results of previous queries (PHI, PII, Finance).

👉 **Session Tokens:** Logical session IDs for hijacking.

👉 **Infrastructure Keys:** AWS Access Keys, Stripe API keys loaded in config.

#4 The Kill-Chain

Step-by-Step Exploitation

#4.1 The Kill-Chain

Step 1: Reconnaissance & Connection

👉 **Targeting:** Port 27017.

👉 **Protocol Deviation:**

👉 Normal drivers send hello handshake.

👉 Exploit scripts are rude.

👉 **Action:** Skip handshake, immediately send malicious OP_COMPRESSED.

#4.2 The Kill-Chain

Step 2: Heap Grooming

👉 **Objective:** Ensure the "dirty" memory contains sensitive data.

👉 **Technique:**

👉 Open **50 concurrent connections.**

👉 49 perform **legitimate logins/queries** (fill the heap).

👉 1 triggers the **exploit.**

👉 **Result:** The allocator **recycles pages** containing **fresh credentials** for the **exploit buffer.**

#4.3 The Kill-Chain

Step 3: Extraction (The Bleed)

👉 **Execution:**

👉 Header: uncompressedSize: 1048576

👉 Payload: zlib("A")

👉 **Outcome:**

👉 Server responds with 1MB buffer.

👉 Attacker captures the stream.

👉 **Analysis:** Grep for SCRAM-SHA-256, {"user":, AWS_ACCESS_KEY.

#4.4 The Kill-Chain

Step 4: RCE Potential

👉 **Primary Impact:** Information Disclosure.

👉 **Force Multiplier:**

👉 Leaked pointers defeat ASLR (Address Space Layout Randomization).

👉 Leaked session tokens allow Auth Bypass.

👉 **Advanced Actors:** Combine MongoBleed with heap overflow to achieve full RCE.

#5 The Fix.

Detailed Remediation

#5.1 The Fix.

```
// The Vulnerable Logic (message_compressor_zlib.cpp)
// It trusted the buffer capacity, not the write result.
Status swM = zlib::inflate(input, &output);
// FATAL: Returns the allocated size (e.g. 1MB), not the data size.
return output.length();

// ------------------------------------------------------

// The Patched Logic
// We explicitly track how many bytes zlib actually wrote.
size_t bytesWritten;
Status swM = zlib::inflate(input, &output, &bytesWritten);

// Resize the buffer to match ONLY the valid data.
output.resize(bytesWritten);
return output.length();

// Result:
// "Slack space" is truncated.
// No uninitialized memory is returned to the client.
// CVE-2025-14847 is neutralized.
```

#5.2 The Fix.

Strategy

👉 Immediate Patching:

👉 Upgrade binaries (8.0.17+, 7.0.28+, 6.0.27+).

👉 Use Rolling Upgrades for Replica Sets.

👉 Workaround (Legacy):

👉 Edit mongod.conf.

👉 Remove zlib from net.compression.compressors.

👉 Detection:

👉 Velocity > 100 conns/min.

👉 Zero Metadata Rate (Connections without Client Metadata).

#6 Developer's Takeaway

Know your functions

#6.1 Developer's Takeaway

Memory Safety is not solved

👉 **Foundations matter:** The internet runs on C++.

👉 **Trust No One:** Never trust a client's length declaration for memory allocation without verifying the actual data processed.

👉 **Know your functions:** The developers thought output.length() returns the actual size. That caused the vulnerability.

👉 **Defense in Depth:** Port 27017 should never be exposed. Internal Trust is a myth.

Status

Patched (Dec 25, 2025). Upgrade to 8.0.17+, 7.0.28+. Disable Zlib if patching is impossible.

#! Anatomy of a Bug

Technical Credits:

Joe DeSimone, MongoDB Sec Team

Author: @tralsesec

#HeapGrooming #Infosec #MongoDB
#ZeroDay #Zlib #Databases
#HeapOverflow