INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-07-08 07:05 UTC

Rocket.Chat File Access Flaw via MongoDB ObjectId Prediction

CVE VULNERABILITY | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CVE-2026-0397
Type	CVE Vulnerability
CVE ID	CVE-2026-30833
Severity	HIGH
CVSS Base Score	7.5
EPSS Score	0.0027 (18th percentile)
Affected Products	Rocket.Chat (version(s) unconfirmed pending NVD advisory review)
Published	2026-07-06
Discovery Source	Gemini

Executive Summary

A file access vulnerability in Rocket.Chat, tracked as CVE-2026-30833, allows attackers to predict MongoDB ObjectId values and potentially enumerate or access files that should be restricted. The flaw was reported by Aikido Security and affects organizations running Rocket.Chat as their internal or customer-facing communications platform. Affected version ranges have not yet been confirmed by NVD; organizations should treat any production Rocket.Chat deployment as potentially exposed until vendor guidance is published.

Technical Analysis

CVE-2026-30833 affects Rocket.Chat and stems from predictable MongoDB ObjectId generation. MongoDB ObjectIds are 12-byte values composed of a 4-byte Unix timestamp, 5 bytes of random machine/process identifier, and a 3-byte incrementing counter. Under certain conditions (e.g., when the machine/process identifier is known or guessable, and the counter is observable), this structure makes ObjectIds partially predictable, enabling an unauthenticated or low-privileged attacker to enumerate sequential or near-sequential identifiers and access file resources without proper authorization. Classified under CWE-284 (Improper Access Control) and CWE-340 (Generation of Predictable Numbers or Identifiers), the vulnerability maps to MITRE ATT&CK T1083 (File and Directory Discovery) and T1078 (Valid Accounts). The NVD-assigned CVSS base score is 7.5 (High). EPSS score is 0.00268 (18th percentile), indicating limited exploitation activity observed to date; the vulnerability is not on CISA's Known Exploited Vulnerabilities catalog. Affected version ranges remain

unconfirmed pending full NVD advisory publication. Source confidence is LOW, primary technical detail derives from a single Aikido Security blog post; independent corroboration is not yet available. Patch status and vendor-confirmed remediation guidance are pending Rocket.Chat's formal disclosure.

Action Checklist

1. Step 1: Containment, Identify all production Rocket.Chat instances in your environment (reference CIS 1.1: Establish and Maintain Detailed Enterprise Asset Inventory). Restrict unauthenticated and low-privileged external access to Rocket.Chat file endpoints at the network perimeter until patch status is confirmed. If Rocket.Chat is internet-facing, consider placing it behind an authenticated reverse proxy or restricting file-access API routes via firewall rule (CIS 4.4: Implement and Manage a Firewall on Servers).
2. Step 2: Detection, Review Rocket.Chat application logs and MongoDB access logs for sequential or near-sequential ObjectId enumeration patterns, specifically, high-frequency read requests against file or upload endpoints from single source IPs or unauthenticated sessions. Enable audit logging across the Rocket.Chat application tier per NIST AU-2 (Event Logging) and AU-12 (Audit Record Generation). Correlate with NIST AU-6 (Audit Record Review, Analysis, and Reporting) to flag anomalous file access sequences. No confirmed IOC patterns or signatures are currently available from the source material.
3. Step 3: Eradication, Monitor Rocket.Chat's official release channel and NVD advisory page for CVE-2026-30833 for a patched version. Apply the vendor-issued patch immediately upon release to all instances. Per Rocket.Chat documentation (source: https://docs.rocket.chat/docs/mongodb-configuration), review MongoDB deployment configuration to assess whether ObjectId generation can be hardened or file access routes can be restricted at the database layer. Once affected version ranges are confirmed by NVD, prioritize patching based on your installed versions.
4. Step 4: Recovery, After patching, validate that file access endpoints require authentication and enforce access control checks that do not rely solely on ObjectId knowledge. Confirm MongoDB audit logging is active and baseline file-access patterns have returned to normal. Review NIST AC-3 (Access Enforcement) compliance for the Rocket.Chat deployment, access to file resources should be enforced by authorization policy, not obscurity of identifiers. Run CIS 7.1 (Establish and Maintain a Vulnerability Management Process) review to confirm the patched version is reflected in your asset inventory.
5. Step 5: Post-Incident, Conduct a broader review of internal applications that rely on MongoDB ObjectIds as implicit access-control tokens. Implement NIST AC-6 (Least Privilege) to ensure file and resource access requires explicit authorization checks independent of identifier predictability. Document this gap in your risk register and update detection rules to flag high-frequency sequential resource enumeration attempts. Review NIST AC-2 (Account Management) and AU-2 (Event Logging) as countermeasures to reduce exposure from low-privileged account abuse in similar patterns.

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate immediately to legal and data protection officer if MongoDB audit logs or Rocket.Chat access logs confirm that unauthenticated or low-privileged sessions successfully retrieved file ObjectIds corresponding to sensitive content (PII, PHI, or confidential business communications), triggering breach notification obligations under GDPR, HIPAA, or applicable state law.

Recovery Notes	After applying the vendor patch for CVE-2026-30833, perform authenticated and unauthenticated curl tests against Rocket.Chat's /ufs/ and /file-upload/ endpoints to confirm server-side authorization enforcement is active and not reliant solely on ObjectId opacity. Monitor MongoDB audit logs and web server access logs for continued sequential ObjectId enumeration patterns for a minimum of 14 days post-patch, as threat actors who identified the window may return to verify whether access was revoked. Confirm the patched version string is recorded in the asset inventory and that any compensating firewall rules applied during containment are reviewed for removal or retention based on the organization's ongoing exposure posture.
Forensic Artifacts	Rocket.Chat web server (nginx/Caddy) access logs: look for GET requests to paths matching /ufs/GridFS:Uploads// from single source IPs with high request frequency and sequential ObjectId values in the counter byte segment — this is the direct fingerprint of ObjectId enumeration exploitation | MongoDB audit log (auditLog.json): `find` operation records against the `rocketchat_uploads` and `rocketchat_room_files` collections, filtered for operations not originating from the Rocket.Chat application service account — unauthenticated or unexpected client IPs performing bulk reads indicate exploitation | MongoDB oplog (`local.oplog.rs`): historical read operation patterns against file-related collections during the suspected exploitation window, accessible even if audit logging was not enabled at the time — query with `db.oplog.rs.find({op:'q', ns:/rocketchat_uploads/}).sort({ts:-1})` | Rocket.Chat application logs (PM2 logs at /home/rocketchat/.pm2/logs/ or container stdout): HTTP 200 responses to file endpoint requests from unauthenticated sessions or sessions with no legitimate file-access history, which would indicate successful unauthorized file retrieval via predicted ObjectIds | Memory dump of the Rocket.Chat Node.js process (captured via gcore before patching): may contain in-memory session tokens, active request state, and cached MongoDB query results that identify which ObjectIds were accessed and by which sessions during the exploitation window

Per-Action IR Details

Step 1: Containment — Identify all production Rocket.Chat instances in your environment (reference CIS 1.1: Establish and Maintain Detailed Enterprise Asset Inventory). Restrict unauthenticated and low-privileged external access to Rocket.Chat file endpoints at the network perimeter until patch status is confirmed. If Rocket.Chat is internet-facing, consider placing it behind an authenticated reverse proxy or restricting file-access API routes via firewall rule (CIS 4.4: Implement and Manage a Firewall on Servers).

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy

Controls: CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 4.4 (Implement and Manage a Firewall on Servers), NIST AC-4 (Information Flow Enforcement)

Compensating: Use iptables or nftables to block external access to Rocket.Chat file-serving routes (commonly /file-upload/ and /ufs/ path prefixes on the Rocket.Chat port, default 3000/tcp) from non-trusted source CIDRs: `iptables -I INPUT -p tcp --dport 3000 -m string --string '/ufs/' --algo bm -j DROP`. For teams behind nginx, add `location ~* ^/ufs/ { deny all; }` to the Rocket.Chat vhost until patch status is confirmed. Enumerate all Rocket.Chat instances with a network scan: `nmap -p 3000 --open 10.0.0.0/8`.

Evidence: Before restricting network access or modifying firewall rules, capture current active network connections to the Rocket.Chat host: `ss -tnp | grep 3000` or `netstat -anp | grep 3000`. Export live connection state showing any source IPs actively accessing file endpoints. Capture Rocket.Chat application logs (default path: `/home/rocketchat/.pm2/logs/` or container stdout, depending on deployment) and MongoDB connection logs (`/var/log/mongodb/mongod.log`) in their current unmodified state before ACL changes alter incoming traffic patterns. Hash all captured log files with SHA-256 immediately after collection to establish forensic integrity.

Step 2: Detection — Review Rocket.Chat application logs and MongoDB access logs for sequential or near-sequential ObjectId enumeration patterns — specifically, high-frequency read requests against file or upload endpoints from single source IPs or unauthenticated sessions. Enable audit logging across the Rocket.Chat application tier per NIST AU-2 (Event Logging) and AU-12 (Audit Record Generation). Correlate with NIST AU-6 (Audit Record Review, Analysis, and Reporting) to flag anomalous file access sequences. No confirmed IOC patterns or currently available from the source material.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis

Controls: NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 8.2 (Collect Audit Logs)

Compensating: Parse Rocket.Chat nginx or Caddy access logs for sequential ObjectId enumeration using a bash pipeline: `grep -E '/(ufs|file-upload)/[a-f0-9]{24}' /var/log/nginx/access.log | awk '{print $1, $7}' | sort | uniq -c | sort -rn | head -50`. MongoDB ObjectIds are 24-character hex strings encoding a 4-byte timestamp, 5-byte random value, and 3-byte incrementing counter — sequential enumeration will show ObjectIds with incrementing counter bytes from a single IP within a short time window. Enable MongoDB audit log if not active: set `auditLog.destination: file` and `auditLog.filter` for `find` operations on the `rocketchat_uploads` collection in `mongod.conf`. Use a Sigma rule targeting web server logs for high-frequency hits on file-serving paths from a single IP within a 60-second window.

Evidence: Before enabling or modifying audit logging settings (which restarts mongod and may flush volatile state), capture: (1) current MongoDB in-memory operation list via `db.currentOp()` in mongosh to identify any active enumeration queries against the `rocketchat_uploads` or `rocketchat_room_files` collections; (2) Rocket.Chat application log tail showing recent /ufs/ and /file-upload/ GET requests with source IPs and HTTP response codes; (3) MongoDB oplog entries (`use local; db.oplog.rs.find({op:'q', ns:/rocketchat_uploads/}).sort({ts:-1}).limit(100)`) to identify recent bulk read patterns against file collections. These volatile query states are lost upon mongod restart.

Step 3: Eradication — Monitor Rocket.Chat's official release channel and NVD advisory page for CVE-2026-30833 for a patched version. Apply the vendor-issued patch immediately upon release. Per Rocket.Chat documentation (source: docs.rocket.chat/docs/mongodb-configuration), review MongoDB deployment configuration to assess whether ObjectId generation can be hardened or file access routes can be restricted at the database layer. Affected version ranges are not yet confirmed — apply any released fix regardless of your installed version until scope is clarified.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Before patching, snapshot the Rocket.Chat MongoDB database: `mongodump --db rocketchat --out /backup/rocketchat-pre-patch-$(date +%Y%m%d)/`. To harden MongoDB ObjectId-based file access at the database layer without a patch, create a MongoDB role that restricts direct `find` access on `rocketchat_uploads` to the application service account only, preventing lateral ObjectId enumeration from lower-privilege connections: `db.createRole({role:'rcFileReadOnly', privileges:[{resource:{db:'rocketchat',collection:'rocketchat_uploads'},actions:['find']}],roles:[]})`. Monitor the Rocket.Chat GitHub releases page (github.com/RocketChat/Rocket.Chat/releases) and NVD entry for CVE-2026-30833 for patch availability.

Evidence: Before applying the vendor patch (which will modify application binaries and may restart MongoDB), capture: (1) full memory dump of the Rocket.Chat Node.js process (`gcore $(pgrep -f rocketchat)`) to preserve any in-memory state showing active sessions or enumeration activity; (2) `netstat -anp | grep 3000` and `netstat -anp | grep 27017` to document active connections to both the app tier and MongoDB at patch time; (3) a final export of Rocket.Chat application logs and MongoDB audit logs covering the exposure window. Verify SHA-256 hashes of the installed Rocket.Chat application bundle before patching to establish a pre-patch baseline for comparison. All captures must precede the patch application — package updates will overwrite application files.

Step 4: Recovery — After patching, validate that file access endpoints require authentication and enforce access control checks that do not rely solely on ObjectId knowledge. Confirm MongoDB audit logging is active and baseline file-access patterns have returned to normal. Review NIST AC-3 (Access Enforcement) compliance for the Rocket.Chat deployment — access to file resources should be enforced by authorization policy, not obscurity of identifiers. Run CIS 7.1 (Establish and Maintain a Vulnerability Management Process) review to confirm the patched version is reflected in your asset inventory.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AC-3 (Access Enforcement), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), NIST AU-6 (Audit Record Review, Analysis, and Reporting)

Compensating: Validate post-patch access enforcement manually: using curl, attempt unauthenticated access to a known Rocket.Chat file ObjectId URL (`curl -v http://:3000/ufs/GridFS:Uploads//`) and confirm a 401 or 403 response — any 200 response indicates the fix has not applied. Confirm MongoDB audit logging is capturing `find` operations on `rocketchat_uploads` by running a test query and verifying it appears in `/var/log/mongodb/auditLog.json`. Update your asset inventory (CIS 1.1) with the patched version string, confirmed by `curl http://localhost:3000/api/v1/info | python3 -m json.tool | grep version`.

Evidence: This step follows patching and does not alter live system state in a way that destroys forensic evidence — prior volatile captures (memory, connections, logs) should already have been completed in Step 3. For recovery validation, document: (1) post-patch Rocket.Chat version string; (2) curl test results for unauthenticated file endpoint access (save full HTTP response headers); (3) a MongoDB audit log snapshot confirming `find` operations on `rocketchat_uploads` are now gated by authentication. Retain all pre-patch evidence per your organization's retention policy for potential regulatory review.

Step 5: Post-Incident — Conduct a broader review of internal applications that rely on MongoDB ObjectIds as implicit access-control tokens. Implement NIST AC-6 (Least Privilege) to ensure file and resource access requires explicit authorization checks independent of identifier predictability. Document this gap in your risk register and update detection rules to flag high-frequency sequential resource enumeration attempts. Reference D3-UAP (User Account Permissions) and D3-LAM (Local Account Monitoring) as countermeasures to reduce exposure from low-privileged account abuse in similar patterns.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity

Controls: NIST AC-6 (Least Privilege), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Enumerate all internal applications using MongoDB by querying your asset inventory or running `ss -tnp | grep 27017` across internal hosts to identify services connecting to MongoDB. For each identified app, review whether resource identifiers (ObjectIds, UUIDs) are used as sole access-control gates — this is the architectural anti-pattern exposed by CVE-2026-30833. Write a persistent Sigma detection rule targeting web server logs for sequential 24-character hex identifier enumeration across any file-serving endpoint: pattern `[a-f0-9]{24}` with >20 unique values from one IP within 60 seconds. Add this detection to a cron-driven log analysis job (`grep -oE '[a-f0-9]{24}' /var/log/nginx/access.log | sort | uniq -c | sort -rn`) if no SIEM is available. Document findings in the risk register with reference to the ObjectId predictability class of vulnerability.

Evidence: This phase does not alter live system state. Compile a final evidence package including: (1) all pre-patch and post-patch log exports with SHA-256 hashes; (2) list of source IPs that accessed Rocket.Chat file endpoints during the exposure window, extracted from nginx/Caddy access logs; (3) MongoDB audit log records for `find` operations on `rocketchat_uploads` during the exposure window, showing which ObjectIds were queried and by which authenticated (or unauthenticated) sessions. This package supports any regulatory breach notification assessment if sensitive files were confirmed accessed.

Detection Guidance

Monitor Rocket.Chat application and web server access logs for high-frequency GET requests targeting file or upload endpoints, particularly patterns showing sequential or incrementally varying ObjectId values in request parameters or URL paths. Flag unauthenticated sessions generating more than a threshold number of file-access requests within a short window. In MongoDB logs, look for sequential _id field queries against collections storing file metadata or attachments. NIST AU-2 (Event Logging) and AU-6 (Audit Record Review, Analysis, and Reporting) provide the control baseline for this monitoring. CIS 8.2 (Collect Audit Logs) should be validated to confirm Rocket.Chat and MongoDB logging is active and centralized. No confirmed IOC hashes, IPs, domains, or signatures are available from current source material; detection relies on behavioral pattern analysis until vendor guidance is published.

Framework Mappings

MITRE-ATTACK

- **T1083** — File and Directory Discovery
- **T1078** — Valid Accounts

NIST-800-53R5

- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **AC-3** — Access Enforcement

OWASP-TOP10-2021

- **A01:2021** — Broken Access Control

CIS-V8

- **6.1** — Establish an Access Granting Process
- **6.2** — Establish an Access Revoking Process

SOC2-TSC

- **CC6.1** — The entity implements logical access security software, infrastructure, and architectures over protected information assets
- **CC9.2** — Manages risks associated with vendors and business partners

HIPAA-SECURITY

- **164.312(a)(1)** — Access Control

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities
- **A.5.21** — Managing information security in the ICT supply chain

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1083	File and Directory Discovery	Discovery
T1078	Valid Accounts	Defense-Evasion

Sources

Source	URL	Tier
Configure MongoDB for Rocket Chat Secure Deployment	https://docs.rocket.chat/docs/mongodb-configuration	T3
CVE-2026-30833 Detail - NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-30833	T1
Predicting MongoDB ObjectId() continuously in Rocket.Chat	https://www.aikido.dev/blog/predicting-mongodb-objectid-rocket-chat	T3
Upgrading Rocket.Chat Docker Compose to MongoDB 8.2	https://forums.rocket.chat/t/action-required-docker-compose-moving-...	T3
Broken RocketChat server after update snap package from 3.18.7 to ...	https://forums.rocket.chat/t/broken-rocketchat-server-after-update-...	T3

Generated 2026-07-08 07:05 UTC by TJS Security Command Center