0 Day Discovery Report

Information Disclosure Observed During Deserialization RCE PoC Work

Author: Daniel C. Ulman (for Carson Saint)
Date of discovery: April 2024

Confidentiality / Ownership

This finding was documented under a binding Non-Disclosure Agreement (NDA). This document contains only information permitted by the mutual NDA.

Executive Summary

In April 2024, during development of a proof-of-concept for deserialization-based remote code execution (RCE) in BentoML, an unauthenticated information disclosure was observed on HTTP endpoints associated with deserialization and runner functionality.
Server responses contained internal version identifiers and configuration strings, including runner and BentoML version markers as well as internal path details. Although no exploit was developed from this condition, the disclosure significantly improved fingerprinting accuracy and potential targeting of deserialization RCE attacks, giving the behavior zero-day characteristics.

Scope & Context

• Engagement: Controlled internal testing performed during deserialization RCE PoC development for Carson Saint.
• Platform: Services handling BentoML pickle, runner, or serde payloads in a test environment.
• Access: Unauthenticated HTTP access within a closed harness; no external systems were targeted.

Technical Findings

• Certain BentoML pickle, runner, and serde endpoints (examples: /runner/submit, /runner/args, /api/serde) returned diagnostic data when triggered with specific markers or malformed requests.
• Response data included internal version identifiers and configuration paths (examples: runner_version, bentoml_version, boe.version, java.version, and filesystem references).
• The issue resulted from verbose diagnostic output and insufficient sanitization in exception handling.
• No memory corruption or privilege escalation occurred — the disclosure was a logic and configuration weakness.

Proof of Concept (High-Level Summary)

• The condition was reproduced using a controlled request marker to trigger diagnostic responses in the isolated test harness.
• Returned data confirmed exposure of internal version and configuration information.
• Testing was limited to observation and verification; no exploit or live deserialization was performed.

Impact

• Primary risk: Exposure of internal software and configuration data that aids precise exploit targeting.
• Severity: High when combined with deserialization RCE, as the disclosure enables accurate version and path fingerprinting.
• Likelihood: Moderate in deployments where diagnostic endpoints or debug logging remain enabled.

Detection Probe (Sanitized Example)

The following snippet represents the single sanitized HTTP request used within the test harness. All other SAINT scanner logic and supporting code are restricted under NDA.

```
// HTTP request formatting (snprintf only — other code is restricted)
snprintf(data, HTTP_OUTBUF_LEN,
    "POST /runner/submit HTTP/1.0\r\n"
    "Host: %s\r\n"
    "Content-Type: application/octet-stream\r\n"
    "Content-Length: 4\r\n"
    "\r\n"
    "ping",
    socktarget);
```

(Note: This block formats the HTTP request only. Detection logic, control-case handling, and match conditions are internal to Carson Saint and omitted here.)

Remediation & Outcome

• The condition was verified in a test environment and reported to the Carson Saint CTO.

Recommended Mitigation

Disable or restrict diagnostic and debug endpoints in production.
Remove or sanitize version and configuration strings from all responses.
Validate and sanitize all reflected data in HTTP responses.
Use schema-based or sandboxed deserialization methods instead of unsafe generic mechanisms.
Implement monitoring to detect diagnostic responses containing internal version strings.