Bugsnag Error Fix: “Cannot read properties of null (reading ‘digest’)”

Issue Summary

Error: `TypeError: Cannot read properties of null (reading 'digest')`
Location: `node_modules/next/dist/compiled/next-server/app-page.runtime.prod.js`
Timestamp: Jan 5th 05:28:29 UTC
Severity: Unhandled error in production

Root Cause Analysis

The error was caused by **incorrect Bugsnag package usage on the server side**:

1. **Wrong Package**: `src/lib/bugsnag-server.ts` was importing `@bugsnag/js` (browser library) instead of `@bugsnag/node` (Node.js/server library)

2. **Error Object Handling**: The code was using type assertions (`error as Error`) without verifying that the caught error was actually an Error instance

3. **Next.js Error Digest**: When Next.js tried to process non-Error objects (null, undefined, or improperly formatted errors), it would fail when trying to read the `digest` property, which is used for error tracking

What Was Fixed

1. Updated Bugsnag Server Library

File: `src/lib/bugsnag-server.ts`

Before:

```
import Bugsnag from '@bugsnag/js';
```

After:

```
import Bugsnag from '@bugsnag/node';
```

Why: `@bugsnag/js` is designed for browser environments and may not handle server-side errors properly. `@bugsnag/node` is specifically designed for Node.js/server environments.

2. Enhanced Error Validation in onError Handler

Added:

```
onError: (event) => {
  // Ensure we have a valid error object
  if (!event.errors || event.errors.length === 0) {
    console.warn('Bugsnag received event with no errors');
    return false;
  }
  // ... rest of handler
}
```

Why: This prevents Bugsnag from processing events that don't contain valid error objects, which could lead to downstream issues with Next.js error handling.

3. Improved Error Normalization

Updated Function: `notifyBugsnagServer()`

Before:

```
export function notifyBugsnagServer(error: Error, metadata?: Record<string, any>) {
  const server = getBugsnagServer();
  if (server) {
    server.notify(error, (event) => {
      // ...
    });
  }
}
```

After:

```
export function notifyBugsnagServer(error: Error | unknown, metadata?: Record<string,
any>) {
  const server = getBugsnagServer();

  // Ensure we have a proper Error object
  const errorObj = error instanceof Error
    ? error
    : new Error(String(error || 'Unknown error'));

  if (server) {
    server.notify(errorObj, (event) => {
      if (metadata) {
        event.addMetadata('custom', metadata);
      }
      // Track non-Error objects for debugging
      if (!(error instanceof Error)) {
        event.addMetadata('debug', {
          originalErrorType: typeof error,
          originalErrorValue: error,
        });
      }
    });
  }
}
```

Why: This ensures that even if code throws a non-Error object (string, null, undefined, etc.), Bugsnag receives a proper Error instance, preventing the Next.js "digest" error.

4. Enhanced withBugsnagServerError Middleware

Updated:

```
export function withBugsnagServerError<T>(
  handler: (req: any, res: any) => Promise<T>
) {
  return async (req: any, res: any) => {
    try {
      return await handler(req, res);
    } catch (error) {
      // Ensure we have a proper Error object before notifying Bugsnag
      const errorObj = error instanceof Error
        ? error
        : new Error(String(error || 'Unknown error'));

      notifyBugsnagServer(errorObj, {
        request: {
          url: req.url,
          method: req.method,
          headers: req.headers,
        },
        originalError: error,
      });
      throw errorObj;
    }
  };
}
```

Why: API route errors are now properly normalized before being reported to Bugsnag, preventing any non-Error objects from reaching Next.js error handlers.

Technical Details

Why This Error Occurs

Next.js internally uses error “digests” to track and identify errors. When an error object is processed, Next.js expects it to have certain properties. If it receives `null` or an improperly formatted error object, it will fail with:

```
Cannot read properties of null (reading 'digest')
```

Package Differences

Feature	@bugsnag/js	@bugsnag/node
Environment	Browser	Node.js/Server
Error Context	Browser console, window, DOM	Process, server requests
Stack Traces	Browser stack traces	Server stack traces
Async Handling	Browser promises	Node.js async/await, call-backs

Testing

Build Test

```
npm run build
```

✅ **Result:** Build succeeded with no errors

Expected Behavior

- Server-side errors will now be properly captured by Bugsnag using the correct Node.js library
- All errors will be normalized to proper Error instances before being reported
- Next.js will no longer encounter null errors when processing error events
- Error metadata will include debug information about non-Error objects that were caught

Deployment

Files Modified

1. `src/lib/bugsnag-server.ts` - Updated to use @bugsnag/node and improved error handling

Dependencies

- `@bugsnag/node` - Already installed in package.json
- No new dependencies required

Environment Variables

No changes required. Uses existing `NEXT_PUBLIC_BUGSNAG_API_KEY`.

Monitoring

After Deployment

1. ✅ Monitor Bugsnag dashboard for the “Cannot read properties of null” error - should not appear anymore
2. ✅ Check server logs for proper Bugsnag initialization: “✅ Bugsnag server initialized successfully”
3. ✅ Verify errors are still being reported to Bugsnag (different errors, not the digest error)

4. ✅ Check for any debug metadata about non-Error objects being caught

Success Indicators

- No more “Cannot read properties of null (reading ‘digest’)” errors in Bugsnag
- Server errors properly tracked with full stack traces
- Error metadata includes server context (Node version, environment, etc.)

Prevention

Best Practices Implemented

1. **Always use proper error packages**: Browser packages for client, Node packages for server
2. **Validate error objects**: Check `instanceof Error` before passing to error handlers
3. **Normalize errors**: Convert non-Error objects to proper Error instances
4. **Add debug metadata**: Track original error types for debugging

Code Review Checklist

- [ ] Verify correct package imports (@bugsnag/node for server, @bugsnag/js for client)
- [ ] Check error handling normalizes to Error instances
- [ ] Ensure error handlers don’t throw null or undefined
- [ ] Validate error objects before passing to third-party libraries

Commit Information

- **Branch:** main
- **Files Changed:** 1
- **Lines Added:** ~40
- **Lines Removed:** ~10

Related Issues

- Bugsnag Error: “Cannot read properties of null (reading ‘digest’)”
- Audit Logs Viewer deployment (feature that was deployed when error first appeared)

References

- Bugsnag Node.js Documentation (https://docs.bugsnag.com/platforms/javascript/node/)
- Next.js Error Handling (https://nextjs.org/docs/app/building-your-application/routing/error-handling)
- Next.js Error Digest Documentation (https://nextjs.org/docs/app/api-reference/file-conventions/error)