GunP4ng / **README.md**

Last active 4 days ago

Code Revisions 3

CVE-2025-70067

README.md

CVE-2025-70067

> Public reference for CVE-2025-70067 (Issue: heap-based buffer overflow in Assimp FBX material property handling).

Product / Affected Versions

Vendor: Assimp project Product: Assimp Open Asset Import Library Affected version(s): Assimp 6.0.2 as tested/reported Fixed version: TBD/Unknown Component(s): FBX importer material parsing, `aiMaterial::AddBinaryProperty`, `code/Material/MaterialSystem.cpp`, `aiString` fixed-size key buffer (`AI_MAXLEN`)

Summary

Assimp's FBX import path can pass an attacker-controlled material property key into `aiMaterial::AddBinaryProperty`. In the reported vulnerable build, the function records the key length, relies on an `ai_assert` size check, and then copies the key into fixed-size `aiString` storage using `strcpy()`.

Because `ai_assert` is compiled out in Release and RelWithDebInfo builds where `NDEBUG` is defined, the runtime length check is not enforced. A crafted FBX file containing an excessively long material property name can therefore cause a write beyond the fixed-size material-property key buffer.

Vulnerability Type / CWE

Type: Heap-based Buffer Overflow / Improper Input Validation CWE: CWE-122 (Heap-based Buffer Overflow), CWE-787 (Out-of-bounds Write)

Impact

Denial of Service: Yes (a malformed FBX file can crash an application that imports it through Assimp.) Memory corruption: Yes (an overlong property key can overflow fixed-size `aiString` storage.) Potential code execution: Possible in theory, depending on allocator behavior, application context, and runtime hardening. Attack requirements: The target application must open or import an attacker-supplied FBX file using Assimp.

High-level Verification / Reproduction (no weaponized details)

The reporter confirmed this issue with AddressSanitizer in Release / RelWithDebInfo builds where `NDEBUG` is defined. Debug builds may not reproduce the overflow because the `ai_assert` length check remains active.

High-level reproduction conditions:

1. Build Assimp 6.0.2 in a non-debug configuration where `ai_assert` is compiled out.
2. Import a crafted FBX file containing an overlong material property name.
3. Reach the FBX material parsing path that calls `aiMaterial::AddBinaryProperty`.
4. Observe AddressSanitizer reporting a write beyond the fixed-size `AI_MAXLEN` key buffer.

The current local `crashes/final` corpus in this workspace was rematched and does not contain the reproducer for this CVE; those local files reproduce CVE-2025-70070 instead. They should not be cited as evidence for CVE-2025-70067.

Technical Notes / Root Cause (for triage)

Location:

- File: `code/Material/MaterialSystem.cpp`
- Function: `aiMaterial::AddBinaryProperty`

Relevant vulnerable pattern:

```
pcNew->mKey.length = static_cast<ai_uint32>(::strlen(pKey));
ai_assert(AI_MAXLEN > pcNew->mKey.length);
strcpy(pcNew->mKey.data, pKey);
```

The check is assertion-only. In release-style builds the assertion is removed, but the unbounded `strcpy()` remains. If `pKey` is longer than the fixed `aiString` destination capacity, the copy can overflow the heap-allocated material property object.

Mitigation / Fix Guidance

- Replace assertion-only validation with a runtime length check.
- Reject or safely truncate material property keys longer than `AI_MAXLEN - 1`.
- Avoid unbounded string-copy functions for attacker-controlled asset strings.
- Add regression coverage for FBX material properties with overlong property names in release-equivalent builds.

Upstream Fix / References for Triage

Upstream fix commit: TBD/Unknown Upstream pull request: TBD/Unknown Release note status: TBD/Unknown

Credits

Discovered by: TaeYong LEE (GunP4ng)

References

- Assimp project: https://github.com/assimp/assimp