graphite SOS Fund Audit Fix Log

Identified Vulnerabilities

MGR-001: Potential Integer Overflow in Memory Allocator (Elevated)

Fixed in c0e27eb refactored in 7440136

MGR-002: Graphite Builds With the Stack Protector Disabled (Moderate)

Fixed in eec9fe3

MGR-003: Graphite/src/Segment.cpp Constructor Possible Null Pointer Dereference (Moderate)

Fixed in 845127

MGR-004: Graphite/src/Pass.cpp CollisionShift NULL Pointer Dereference & Integer Overflow (Moderate)

The presence of m_collisions is tested in the calling function Pass::runGraphite with:

```
if (!collisions || !m.slotMap().segment.hasCollisionInfo())
        return true;
```

So no action. In addition this covers the other unchecked instances of calls to collisionInfo.

Analysis results in no action

MGR-005: Graphite/src/inc/List.h Possible Integer Overflow (Moderate)

distance() returns a signed integer by design since it's intended to be compatible with the standard C++ vector class. It is up to the callers to ensure that the value is +ve where required. In the case of uses in List.h the callers would fail on the same inputs for other reasons and provide the same guarantees and limitations as the std library versions.

Analysis results in no action

MGR-006: Graphite/src/inc/Rule.h Slotmap::operator[] Does Not Check Bounds (Moderate)

This is by design as slot map is used in performance sensitive code, and follows standard practice for STL C++ collections. All uses should (and are the best of my knowledge) protected by checks occurring immediately before the operator calls or are calling with known safe indexes e.g.[0] or [-1].

Analysis results in no action

MGR-007: Graphite/src/Font.cpp Font::Font() Division-related FPE (Moderate)

This check in Face::readGlyphs()

```
if (e.test(!m_pGlyphFaceCache, E_OUTOFMEM)
|| e.test(m_pGlyphFaceCache->numGlyphs() == 0, E_NOGLYPHS)
|| e.test(m_pGlyphFaceCache->unitsPerEm() == 0, E_BADUPEM))
{
return error(e);
}
```

Happens early enough that it's not possible for a upem of 0 to get through to the Font class constructor.

The remaining case of a upem == 0xffff (since upem is unsigned) and a ppm == INT_MIN is possible only when a crafted font is used in combination with an application calling gr_make_font* calls with a ppm of INT_MIN. To mitigate this commit 99cf79c checks ppm is never <= 0 (a nonsensical value anyway).
Is zero every a legitimate value for pixels per em?

MGR-008: Graphite/src/Font.cpp M_advances NULL Pointer Dereferences (Moderate)

Fixed in 6477dce

MGR-009: Floating Point Exception in VM (Moderate)

Fixed in 924677b

MGR-010: Possible NULL Pointer Dereference in ShiftCollider::mergeSlot() (Moderate)

Fixed as part of MGR-003. Fixed in 64cf8dc

MGR-011: Potential Crash in FileFace::get_table_fn() (Low)

Fixed in ce22348

MGR-012: Potential Use After Free When Logging (Low)

Fixed in df141bf

MGR-013: The LZ4 Parser Does Not Pass the Tests From Liblz4 (Low)

Fixed a bug in lz4::decompressor b10fb35 preventing it from decompressing the last literal block when it started less than 8 bytes from the end of src buffer. This should fix regeneration mismatch errors. Our decompressor places no upper limits on the input block so it will never pass input too large tests by design.

MGR-014: Incomplete Sanity Check When Looking up Glyphs (Low)

Restructured the test in GlyfLookup and added a check in CheckTable. The test in GlyfLookup is now:

```
if (nGlyfOffset + pByte < pByte || nGlyfOffset >= nTableLen - sizeof(Sfnt::Glyph))
return NULL;
```

with a check in CheckTable that nTableLen >= sizeof(Sfnt::Glyph) s.t. nTableLen - sizeof(Sfnt::Glyph) therefore is always positive.

Fixed in a7c4a05

MGR-015: Graphite/src/inc/Compression.h::overrun_copy Integer Overflow Leads to Uninitialized Buffer (Low)

Checked at call site to ensure e does not overflow (presumably we're talking about (e = s+n)).

Analysis results in no action

MGR-016: Graphite/src/inc/Compression.h::overrun_copy Possible Buffer Overflow (Low)

The observed behaviour is intentional (hence the name overrun_copy) and it is up to the call sites to ensure any invocation is safe, I'm not sure given the checks occur immediately before the calls with no additional processing or function calls what could change between the check and use.

Analysis results in no action

MGR-017: Graphite/src/Segment.cpp linkClusters Null Pointer Dereference (Low)

Fixed in 64cf8dc

MGR-018: Graphite/src/inc/Sparse.h Sparse(x,y) Code Smell (Low)

Change committed to master since not a security bug per se. Fixed in 2f0d83b in public master

MGR-019: Graphite/src/inc/FeatureMap.h Possible NULL Pointer Dereference (Low)

Fixed in 77da521

MGR-020: Graphite/src/Code.cpp Machine::Code::Code() Constructor Possible Memory Leak (Low)

Fixed in f28331a

MGR-021: Graphite/src/inc/List.h Possible Integer/memory Overflow (Low)

Fixed in 712e47d