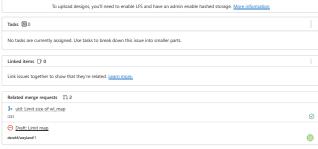
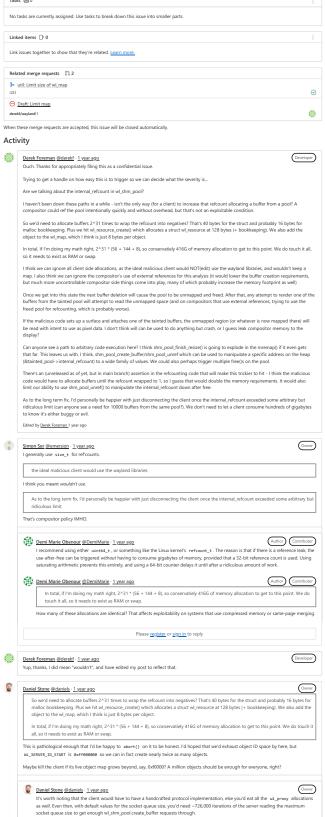
Reference count overflow in shm leads to use-after-free

The shared memory code uses an int for the reference count. On 64-bit systems, a malicous client can create so many references that the int. This is undefined behavior, but it will most likely cause the int to overflow, causing an exploitable use-after-free. A successful exploit will result i execution of arbitrary code in the context of the Wayland compositor.

To fix this bug, wintern t (or intern t) should be used for all reference counts. There are only UINT MAX possible distinct pointers, so a wintern t set reference count never overflow, and an intptr_ reference count cannot overflow if the referencing and referenced objects are at least 2 bytes. This issumes that INTPTR_MAX == (UINTPTR_MAX >> 1) && UINTPTR_MAX >> 1) & UINTP





Developer

Please register or sign in to reply

Maybe kill the client if its live object map grows beyond, say, 0xf0000? A million objects should be enough for everyone, right?

I feel that's reasonable. Also has the added benefit of being far enough removed from refcounting that I'm not certain it would require co-ordinated disclosure to land such a patch?

Daniel Stone @daniels · 1 year ago

There are a couple of automotive environments where the compositor lives across a trust boundary from clients, but apart from the Qubes, I don't believe there are many around for which a compositor compromise would be catastrophic.

I think it's probably still worth a CVF mind, and backporting

[edit: sandboxed clients like Flatpak are in a different privilege domain, and there's also Chro

Edited by Daniel Stone 1 year ago

Demi Marie Obenour @DemiMarie · 1 year ago



I was going to say this does not matter for Oubes OS, but then realized that future (reasonable) enhancements to the GUI protocol could make it exploitable. That said, sandboxed Flatpak applications are certainly less trusted than the compositor, so a CVE is definitely justified.

In practice, I doubt that this is exploitable (beyond a DOS) on most systems, purely due to the amount of memory required. Nevertheless, a client should not be able to crash the compositor, so some sort of resource limit is definitely needed. An ob-limit is a good place to start, but I vounder if finer-grained accounting would be worthwhile. That can certainly come later, it

Finally, as a Qubes OS developer, thank you for considering Qubes OS when analyzing the impact of this bug! Qubes OS uses a custom GUI protocol that is currently backed by X11, but I am working on having it be backed by Wayland instead

Edited by Demi Marie Obenour 1 year ago

Daniel Stone @daniels · 1 year ago



Righto. I'll chase a CVE number through X.Org. Do you want to do the writeup & make sure you're credited for it since you've don the hard work and analysis here (and many thanks for that)?

Demi Marie Obenour @DemiMarie · 1 year ago



Thanks for the incredibly quick response! I receive response (on a weekend, no less!) is refreshing. ! I recently spent several months getting some RPM vulnerabilities natche

When should I send the writeup to distros@vs.openwall.com? Also, while the object map size limit fixes the immediate vulneration When stood sets due metaps to describe the stood of the stood sets due to the stood sets too (now fixed) unlerabilities in the FreeSD kennel that resulted from them. Furthermore, Rust considers failing to call an object's destructor to be a sind most cases, and two duel prefer if this did not lead to use affer-free via refcount overflows. Worst of all, reference leaks do not require gigabytes of memory to exploit. Preventing refcounts from wrapping reduces the impact to denial of service at worst.

Daniel Stone @daniels - 1 year ago



We usually disclose through the cory-security@ list and get CVEs through there to co-ordinate with distros etc. I'm just drafting up email there now and I'll CC you.

I agree with you on refcount hardening, but I think I'd again prefer to impose a stupidly-high limit rather than try to push us up to the full 64-bit boundary. Is the refcount hardening something you'd be interested in working on, or?

Hope you have more exciting things planned for your weekend after this!

Daniel Stone @daniels · 1 year ago



Thinking about it more if you can convince the compositor to hold a ton of references to the pool you could make it happen with fewe objects, va_shm_buffer_ref_pool() exists and asserts that the external + internal refcount stays positive. So I think you'd just' need to get it to take (2 << 31) external refs, then you'd only need a single live client buffer to break the addition test we do in the unref path.

Enlightenment, KWayland, and wiroots all explicitly take a ref on the pool here. But this is harder rather than easier to practically exploit I can't see a way which requires you to do this without at least a live surface object per ref, and these all have larger internal tracking structures for their surfaces than our internal per-buffer structures.

So I think what we need to do is:

- change the (internal_refcount + external_refcount) < 0 test in the pool-unref path to check if either of them are non-zero
- make a new static ref function symmetrical with unreal and change the internal and external refs to use this
 impose an arbitrary limit on the individual refcounts to ensure neither of them climb beyond some silly number which keeps us far from overflow on either or both

Am I on the right path?







Please register or sign in to reply





Demi Marie Obenour @DemiMarie · 1 year ago
You are. What about adding:



#ifdef NDEBUG # error wayland-shm.c must be built with assertions enabled

#endif #include <assert.h>

shm_pool_unref(struct wl_shm_pool *pool, bool external)

if (external) { } else {

}
assert(pool->internal_refcount >= 0 && "internal reference count overflowed");
assert(pool->external_refcount >= 0 && "external reference count overflowed");
if (pool->internal_refcount != 0 || pool->external_refcount != 0)
return;

sap(pool->data, pool->size) == 0 && "munmap() failed - possible memory corruption");

That said, this idea needs work

Edited by Demi Marie Obenour 1 year ago

Derek Foreman @derekf · 1 year ago



I realize this code was already an assert, but maybe it should be wl_abort() instead? On the off chance someone's actually compili assertions.

I think the logic is inverted on the early return? Isn't that the only case in which we want to perform the munmap/free?

Technically, getting into a refcount < 0 case isn't only possible via overflow - a compositor bug could cause the external refcount to go negative. I think that's probably more likely to happen than the overflow case, so the text could be a little confusing

I know some users of this library are very hostile towards assert/abort in general, so I don't know that adding a new one for munmap() failure is going to be popular. I don't see how we can trust anything that occurs after the point of munmap() failure, so I certainly wouldn't complain about it.



Derek Foreman @derekf · 1 year ago

cops, I missed the #ifdef NDEBUG #error. I think I'd prefer to use wl_abort() instead, but don't really have a strong opinion.:)



Pekka Paalanen @pg · 1 year ago

Let's use wl_abort()



Forcing asserts on just makes things more annoying to whoever wants to build stuff.









Please register or sign in to reply Derek Foreman @derekf · 1 year ago I do like <u>distantles</u> suggestion of having a symmetric ref function (static vaid shm_pool_ref(struct wl_shm_pool *pool, bool ex-perhaps) and limiting the combined refocunt before it gets out of hand. Then we'd be asserting exclusively on refounting bugs and overflow bugs in the unref function. (Author) (Contributor) Demi Marie Obenour @DemiMarie · 1 year ago What about something like: #ifdef NDEBUG
error wayland-shm.c must be built with assertions enabled
##endif
##include <assert.h> typedef uint32_t w1_refcount; // no _t because POSIX reserves that namespace wl_ref(wl_refcount *refcount) const wl_refcount count = *refcount;
assert(count < (ULNT32_MAX / 2) &8
"Reference count exceeded UINT32_MAX / 2 - possible reference leak");
*refcount = count + 1;</pre> static void
wl_unref(wl_refcount *refcount) const wl_refcount count = *refcount;
assert(count > 0 && "too many calls to wl_unref");
*refcount = count - 1; static void
shm_pool_unref(struct wl_shm_pool *pool, bool external) } else {
 wl_unref(&pool->internal_refcount); if (pool->internal_refcount != 0 || pool->external_refcount != 0)
 return: assert(munmap(pool->data, pool->size) == 0 && "munmap() failed - possible memory corruption"); free(pool); shm_pool_ref(struct wl_shm_pool *pool, bool external) assert((pool->internal_refcount || pool->external_refcount) &&
 "Calling shm_pool_ref on an already freed pool");
wl_ref(external ? &pool->external_refcount : &pool->internal_refcount); The major advantage of this approach is that unsigned integers are defined to wrap, whereas signed integer overflow is already undefined behavior. Of course, the <code>asserts</code> can be replace with explicit calls to <code>write/fprintf/etc</code> and <code>wl_abort</code>. I am no expert on reference count hardening, but this can be further improved once the issue becmes public ©DemilMadie | Can format this as a path and bring it into my MR for review there if you'd like to forego making you and messing around with the gitlab confidential MR work flow. As usual, we'll need your Signed-off-by: attribution before we can land it Demi Marie Obenour @DemiMarie 1 year ago
That is fine, but be sure to review it first, this was meant as "here is an idea" and not "this can go in straight away."

(Contributor) Please register or sign in to reply Derek Foreman created merge request derekf/wayland!1 (closed) to address this issue 1 year ago Derek Foreman @derekf · 1 year ago eems I have little understanding of how confidential merge requests work, and I'm unsure if anyone here can access my MR... My MR does not address refcounting, which needs to be handled (if just to close the internal+external refcount on the number of elements in a wl_map . Unsure whether it belongs here due to its relevance to this issue, or if I should create a new confidential issue for that as well. Author Contributor Demi Marie Obenour @DemiMarie · 1 year ago I certainly cannot see the MR Daniel Stone @daniels - 1_year ago

[can, but only because I'm an admin. The MR is only on your own repo, not on this one __you'll need to push to a branch in this rimake a confidential MR I think? (You're the first one to do this, congrats.) Derek Foreman @derekf · 1 year ago I certainly cannot see the MR. Disappointing, but unsurprising. Sorry about that! I can, but only because I'm an admin. The MR is only on your own repo, not on this one ... you'll need to push to a branch in this repo to make a confidential MR I think? (You're the first one to do this, congrats.) I think since this is a public repo, anything I push to a branch in it will have immediate visibility, so I don't think that's the way forward: https://docs.gitlab.com/ee/user/project/merge_requests/confidential.html gives me the impression that I must fork the repository, and make my fork private (which I have done). I think I've failed by not creating my fork in the same group or subgroup as the parent public repository, which would share ownership permission? However, that still doesn't seem like what someone would generally want, as it will frequently leave the person who reported the issue out of the loop. The merge request apparently ends up being against my private fork. I think once that's merged I'd have to do a MR from that new branch to the public main branch, and that would immediately (and appropriately) be visible to all. It may be that my best way forward is just giving all participants here visibility into my private fork? Please register or sign in to reply 8 With @daniels help in creating a secure fork, I've put together a merge request that should theoretically have reasonable pe

Daniel Stone made the issue visible to everyone 4 months ago

Please register or sign in to reply

□ <u>Derek Foreman</u> mentioned in merge request 1231 (merged) 7 months ago
 □ <u>Derek Foreman</u> closed via commit <u>b19488c7</u> 6 months ago