chrome.screenlockPrivate - New Chrome API Proposal

Proposal Date - 11 Nov 2013
Primary Eng and PM contacts - benjhayden, joshwoodward
Which team will be responsible for this API? - benjhayden, jcivelli
Also interested - Everybody else who is experimenting with password-free authentication
Target milestone - 33 (branch 17 Dec 2013; stable 25 Feb 2014)

Overview
This API allows Chrome Apps to lock or unlock the screen on ChromeOS, monitor when the screen is locked or unlocked by other means, and show messages to the user if the app decides not to unlock the screen for some reason.
The easier it is to unlock a screen, the more likely it is that a user will lock it in the first place.

Use cases
A platform app may use the USB, NFC, and/or Bluetooth APIs to communicate with a secondary trusted device such as a phone, ring, watch, or badge, thereby allowing that trusted device to serve as an alternative form of authentication for the user.

How would you implement your desired features if this API didn't exist?
The entire app might need to be implemented in C++ in Chrome. The app may implement other features besides screen unlocking, and may be unable to safely share control of USB devices with Chrome.

Can these use cases be addressed by leveraging the standard web platform?
No.

If not, is it something that could/should be part of the web platform?
No. This functionality can be accessed on non-ChromeOS platforms using native messaging. Secure sites and apps already have ways of locking down access and authenticating without blocking access to the rest of the computer.

If not, could these use cases be addressed by extending the functionality of an existing chrome.* API?
The closest related APIs are privacy, identity, power, and sessions, but none look close enough to me.

Does this API expose any functionality to the web?
No.

Do you expect this API to be fairly stable? How might it be extended or changed in the

future?
Apps may eventually want to display other UI in addition to the unlock WebUI besides simple strings, e.g. a swipe pattern matrix.
Apps may eventually want to mediate login as well as unlock.

If multiple apps/extensions used this API at the same time, could they conflict with each other? If so, how do you propose to mitigate this problem?
It would be possible for multiple apps to disagree about whether the screen should be locked or unlocked, and rapidly lock and unlock the screen when they detect that the other app changed the state.
The API could throttle locking and/or disable apps that lock shortly after the screen is unlocked. It is considered worse to lock the screen while the user wants to use it than to unlock the screen while the user may be away.

List every UI surface belonging to or potentially affected by your API:
Apps may show, hide, and show text messages on the unlock (not login) WebUI.
TODO showMessage UI mock

Actions taken with app/extension APIs should be obviously attributable to an app/extension. Will users be able to tell when this new API is being used? How? Can it be spoofed?
At first there will be only one app allowed to use the API, so if the screen unlocks without the user typing their password, then they can know that it was the whitelisted app.
In general, apps should require an unmistakable user gesture, e.g. tapping a badge to an NFC reader, pressing a button on a Bluetooth watch, etc.
I would rather not create a notification when an app unlocks the screen. When the user unlocks the screen, they want to use the computer, not close a dialog then use the computer.

Does this API impose any requirements on the Chrome Web Store ?
No.

Does this API have any interaction with other Chrome APIs ? Does it impose any restrictions on other APIs that can be used in conjunction ?
The screen capture APIs probably cannot capture a screen that is being hidden by the chromeos ScreenLocker. I can't think of any non-obvious restrictions or interactions.

How could this API be abused?
Locking or unlocking the screen when the user didn't intend that.
This API is private and apps that use it are whitelisted. Before whitelisted apps are released, it will be reviewed by security and privacy.

Imagine you're Dr. Evil Extension Writer, list the three worst evil deeds you could commit with your API (if you've got good ones, feel free to add more):

1. Lock the screen whenever it detects that the screen was unlocked.
2. Unlock the screen without requiring an unmistakable user gesture.
3. Monitor when the screen is locked or unlocked and transmit that information to Evil HQ.
4. Show unsavory or misleading messages on the unlock WebUI.

What security UI or other mitigations do you propose to limit evilness made possible by this new API?
Make the API private and whitelist select apps.
Tag messages sent from apps to the unlock WebUI with the app's icon.

Could a consumer of your API cause any permanent change to the user's system using your API that would not be reversed when that consumer is removed from the system?
No.

Draft Manifest Changes
The "screenlockPrivate" permission.

Draft API spec
https://codereview.chromium.org/60583003/diff/60001/chrome/common/extensions/api/screenlock_private.idl

Open questions
Namespace? chrome.screenlockPrivate? chrome.system.screenlock?
Private or public? Web platform?
Support other platforms?