Keeping private data local with device APIs in web environments

Jens de Smit, SURFnet
jens.desmit@surfnet.nl

June 24, 2010

1 Web applications and local resources

A rapid development is happening where many software applications that traditionally are installed and run on end user's machines have now become web applications. Popular applications such as mail clients and financial administration systems that are inherently networked have long since moved to the web. Traditionally more demanding applications such as office suites from Google and Microsoft are now also moving away from the web and getting an online variant. The ubiquitous availability of the web and the fact that online software can take a load off of local resources is an important reason for users to like online software.

Parallel to this another development is taking place. With improvements in technology, video and audio communication software such as Skype, MSN and similar options have caused a general availability of cameras and microphones in or attached to user's machines. This hardware obviously has uses beyond direct communication, such as but not limited to: recording, gesture control, audio commands, image recognition and augmented reality. Modern mobile devices such as smartphones contain even more input devices that can provide local (and therefore personal) information, such as a GPS chip and a compass. Some ideas have also been offered to include biosensors into mobile devices to do, for example, heart rate monitoring of cardiac patients. Recently, there have been propositions in the context of W3C Working Groups (Capture API, Device API) to allow access to these input devices from the context of a web browser.

The combination of these two developments opens up the possibility of advanced web-based applications that heavily utilize these local resources. This opens up enormous possibilities for rapid development and deployment of innovative software that combines more inputs than ever before into a, hopefully, more intuitive and easier to use experience. The same development also opens up very big and very important questions concerning the privacy of the users of this software. How, if at all, is a user's image, voice, location or medical condition guaranteed to be used for just the purpose for which it was recorded and nothing else?

Already, many groups are drafting ways to define, store, transport and enforce policies that govern the retention and replication of such and other personal data. These are great developments and should be given continuous attention, as their implementation will inspire a level of trust in the users of these applications. However, an even higher level of trust can be achieved by guaranteeing that only the information that is required for proper functioning of an application ever reaches the web, and that all other personal data is confined to the local context.

2 Keeping it local

For instance, in an augmented reality marketing campaign users can hold up a fiduciary marker to their webcam in order to view a 3D-model of a product, such as a car. This experience would be perfect to deliver through the web as part of a marketing campaign website and requires the use of a user's webcam to recognize the marker. However, the video feed this camera generates does not need to go anywhere once it has been analyzed for the presence, location and position of the marker. If there were a way for a user to specify that the images from his or her webcam are only allowed to be processed locally, this would greatly enhance the privacy of users. There are multiple approaches to achieving the scenario described above and the following text will explore two such approaches.

2.1 Limited access through abstraction

One way to prevent improper use of device input is to limit what can be done with the device by abstracting away everything but the required functionality. For instance, instead of offering a developer access to a video stream to do marker recognition, the browser implements the recognition and tracking algorithm and offers only access to the output of the algorithm.

The obvious benefit is that this prevents a malicious application that pretends to only do marker tracking but in the meantime also captures the video feed is impossible to build in this model. Additionally, the presence of a marker tracking algorithm and interface greatly simplifies application development, as developers need not implement their own tracking algorithm.

The downside to this approach is that this also limits possible uses of local devices to those that have been defined and implemented upfront. If the standard defines tracking square markers but research later shows that hexagonal markers have obvious advantages over square markers, web developers will have to wait for standard bodies to write an additional specification and then on browser vendors to implement the specification before being able to use the technology. With the number of input devices increasing this definition-implementation delay will expand to a point where innovation and development will be severely hindered.

Additionally, this approach does not provide water tight security. One could imagine that malware authors find a way to define a marker pattern that reliably recognizes hi-fi stereo systems or other expensive items. Combined with the Geolocation API this would give burglars a visual search engine for good burgling spots. Combined with geolocated social network updates to recognize when home owners are on the move this could evolve into an automated dispatch system for smart burglars. This same argument may hold for other uses of local input data: abstracted data may still provide plenty of personal information beyond its obvious use.

2.2 Limited access through locality guarantees

Another way to approach the issue is to declare what may be done with input gathered from local devices. A user could specify that, by default, input gathered from his or her webcam is never allowed to be transmitted over the network or stored locally. Any attempt to transmit data originating from the webcam should be denied as a security violation. This would also need to be enforced for data that is inferred from the input data, such as positional data gathered from a tracking algorithm. Users would define global or per-site policies for restrictions on the use of various local resources similar to how they now define when and where to store passwords. By never allowing, without explicit prior consent, data gathered from local devices to leave the local machine, users have a strong control over what they allow to go online.

Enforcing this method technically will not be trivial. One way would be to mark any variable that has been set with data from a local resource as 'sensitive'. Any variable that has been set or altered under influence of a sensitive variable becomes sensitive as well. Influence in this context could be copying (part of) the variable, using it in a calculation and assigning the result somewhere or using it as a deciding factor in a branching construct such as an `if` or `for` statement. Variables marked as sensitive should then not be allowed to be used as input for any form of network communication.

Another method would be to only grant access to policy-protected resources to isolated execution contexts that only accept input from the main context and never let out any data to prevent leaks. These isolated contexts would then also be denied access to any interface that provides network communication. This latter model requires something of a paradigm shift in traditional client side web programming as it requires the notion of multiple execution contexts and a mechanism to specify how each execution context is allowed to act.

3 Statement of position

The author would like to see if there is support in the W3C community for the concept of enforcing certain data streams to be confined to the local context unless there is explicit user consent for sending the data somewhere else. If there is, there should be an effort to a) define a methodology to specify policies

that are understandable for end users, respectful of user's privacy and enforcable and b) investigate what technique or technology would be capable of enforcing these policies.