WebCrypto Analysis Highlights

Kelsey Cairns and Graham Steel

Abstract

WebCrypto is a controversial new API which specifies cryptographic operations accessible to JavaScript. Here we provide a few highlights from a comprehensive review of this API. Namely, we clarify the intended nature of the API as well as some concerns regarding its functionality within that scope.

We recognize that cryptography for JavaScript is an extremely controversial topic. Many of the concerns and criticisms the W3C met with during the development of the WebCrypto API are valid concerns pertaining to the difficulties of running crypto in the web environment. However, not all criticisms are equal, an often WebCrypto is condemned based on a single example application for which it is simply not a good fit. In our work on WebCrypto, we attempted to provide an unbiased opinion on the security of WebCrypto. Given the depth of our analysis, some things have become apparent to us that may not be obviously at a cursory level. First, there are things that WebCrypto is not meant to be. For example:

- WebCrypto is not an interface intended to be used by the majority of web developers. It is meant to be used either by an experienced security engineer or as an underlying API to future higher-level APIs.
- WebCrypto is not meant to protect your online banking or shopping transactions. It may be used on top of TLS to ensure that information remains encrypted continuously between end point applications. Alternatively, it may be used to impede pervasive monitoring. But insofar as malicious JavaScript is a threat, web crypto is not intended to be the sole protection mechanism for critical data.

The items in this list stem from the fundamental limitation of an API designed to be called during the execution of remote code. Keeping this environment in mind, we do believe that the API should provide as many security properties as it can within its limits. We reviewed the API using a mixture of formal and informal analysis and literature cross-referencing. Upon completion, we do have some concerns about the appropriateness of some of the features given the JavaScript environment.

Our principal concerns are the following:

- It's not clear what security properties, if any, the specification is supposed to ensure. In particular, the key objects have 'attributes' like sign, encrypt, extractable that seem to be intended to control usage, but since there is no specification of a mechanism for persistent storage, it's not clear how or if these properties would be maintained over sessions, and further,

since any key that is 'extractable' can be read in the clear, it's not obvious what purpose restricting the usage of these keys serves.

- Since all functions can be redefined or polyfilled by malicious code running in the same origin, scenarios giving reasonable security properties for the use cases proposed in the specification are rather limited.
- The algorithms proposed include several with known security issues and/or that are deprecated for use in future applications, such as the hash function SHA-1.

Given the time constraints it seems impossible to address these concerns in version 1.0 of the specification (although during the review process the weakest encryption mode specified in the algorithm list was removed). For the next version, we suggest the TC examine the specification of some way to persistently store keys securely such that they retain their attributes, specify a wrap/unwrap mode that uses authenticated encryption with associated data to securely tie the correct attributes to the encrypted key blob in order to enable flexible key management, and investigate a way to make the functions in the cryptographic API immutable by client code.