Testimony to the ONC API Privacy and Security Task Force
Virtual Hearing
Eve Maler, ForgeRock
January 26, 2016

Thank you for the opportunity to address the panel on Consumer Technologies. Following are comments on the topic of APIs and their privacy and security aspects. My background:

Eve Maler is VP of Innovation & Emerging Technology in ForgeRock's Office of the CTO. She is a renowned strategist, innovator, and communicator on digital identity, access, security, privacy, and consent, with particular focus on creating successful interoperable ecosystems and fostering individual empowerment. Eve drives innovation for the ForgeRock Identity Platform and directs ForgeRock's involvement in related industry standards, particularly for federated authorization, privacy, and consent and their impact on the Web and the Internet of Things. To these ends, she founded and chairs the User-Managed Access (UMA) work group and co-founded and co-chairs the Health Relationship Trust (HEART) work group.

Eve was formerly a principal analyst at Forrester Research, advising clients on emerging identity and security solutions, consumer-facing identity, distributed authorization, privacy enhancement, and API security. Previously Eve was an identity solutions architect with PayPal and earlier a technology director at Sun Microsystems, where she co-founded and made major contributions to the SAML federated identity standard. In a previous life she co-invented XML.

APIs as a mechanism for fostering data accessibility

Modern web-based APIs are in fact closer to the traditional definition of a communications protocol. They define a technical "contract" that lets disparate software entities (server and client respectively) communicate with each other, quite often having implemented their respective sides using different programming languages, to achieve a coordinated end, such as transferring data or outsourcing the execution of an algorithm.

The ease with which partner ecosystems can be built to take advantage of this accessibility, along with mobile client apps' need to call APIs in order to function, have led to a robust worldwide "API economy," with many thousands of organizations becoming API providers and working in turn with at least an order of magnitude more client applications. The modern Internet of Things (IoT) era owes its existence largely to the API economy.

In many corporations, money-making APIs have product managers and pricing plans rather than being owned by, say, an enterprise architect and being called by line-of-business client apps.

An “open data” movement has also sprung up to make data sets, particularly government-managed ones, freely available through APIs.

Access control as a counterpoint goal to data accessibility

Some feel that API security and API accessibility are at odds. In my opinion, they need not be.

Traditional IT security practices often give a false sense of security if they present a strong firewall and little resistance thereafter (the classic “crunchy shell and chewy center”). A defense-in-depth approach is more appropriate but often hard to achieve. However, publishing an API to the outside world requires acknowledging that the API’s endpoints (URLs) are indeed on the edge of the network and taking suitable steps.

API management and security solutions, largely built on gateway architectures, have grown up to serve the “API economy” market, including security features to catch malicious types of access, rate-limiting to catch denial-of-service attacks even if the APIs are intended to be open to all, access control to ensure only correct users and applications get through, and other types of access-restricting functionality.

It should be noted that access control is not just for the purpose of security. You can’t charge for an API if you can’t throttle access!

Further, the very nature of an API, versus many other mechanisms for making data accessible – such as enabling FTP (quite often left unsecured) – is a model of constrained access, since the set of supported API calls is typically smaller than the full set of HTTP verbs, with the other calls producing errors.

API design and potential impacts on privacy

Data protection is not privacy, and problems are likely to arise that are out of the realm of the technical.

The best-protected API may still have been designed to be destructive of privacy by virtue of the data its messages carry. For example, the data contained in response data sets could enable inappropriate correlation of individual identities by the receiving party. This could be a problem of API design or data model.

If a legitimate client app must in the course of business be exposed to sensitive or regulated data but the API provider, or the data subject, desires to put other controls on the operator of that app, the challenge enters the business or legal realm. Or if data is transferred only between providers on a “back channel” without any say in the matter by a patient – the way much marketing data changes hands today – then any problems are not the fault of the technology, but rather the business model, and perhaps regulatory compliance. (This notion of parceling out problem spaces has become known as the “BLT sandwich,” for business-legal-technical.)

API challenges and potential solutions in data tagging

APIs present some unique circumstances regarding data tagging to track provenance.

When creating fairly static, non-volatile data, such as asking an individual to fill out a form or recording details of a visit to a healthcare provider, tagging the data creates no problem. But what if an API endpoint is able to report out a live feed of data coming from a device that has a sensor for blood oxygen levels? The most upstream point of provenance is the API, or the device.

A solution would be to identify the points where the API or device is onboarded to its service ecosystem, formalize that onboarding ceremony, and apply security tags to elements of the metadata used in that ceremony.

Industry-standard APIs

Standardizing an API within an industry is a very valuable idea when interoperability – removal of business and technical friction – is needed for some sufficiently large subset of interactions among players. FHIR is one example of where industry movement has been kick-started through a standard API. The Open Bank API effort in the UK is another.

APIs for standard security, identity, and consent mechanisms

For any API, proprietary or standard, there are good reasons for it to use standardized mechanisms for security, identity, and consent to the extent possible. Some reasons:

- Complexity and variation are enemies of security. Standardization simplifies.
- It's hard to separate data from different parts of a person's life, such as "consumer," "health," "car," "travel," "school," and so on. A standard way of identifying the person across those worlds could help bring the data together for their benefit.
- A standard mechanism has likely been well vetted by others.
- Standards can generally be implemented by multiple parties, and those implementations usually strive for interoperability with each other, so it may be more possible to "buy" vs. "build" at a favorable price.

Since APIs are a good technology generally, it stands to reason that they are a good idea when it comes to designing standard mechanisms for security, identity, and privacy as well. This is where the innovative emerging technologies OAuth 2.0, OpenID Connect, and User-Managed Access (UMA) come into play; in part, their specifications include definitions of APIs, and they are extremely well suited for use with APIs.

OAuth is an API of enabling a client app to call an API on behalf of a "resource owner" (typically an individual) and with their consent, without ever having seen their credentials (such as a username and password). An "access token" stands for the consent and for the list of actions ("scopes") the client app can perform, which

may not be the whole possible list. The resource owner can always go back to the API publisher and withdraw the consent, revoking the token.

OAuth is innovative because it ensconces the individual as an intermediary in the API call flow, it protects their credentials, and it provides a means for consent notification. However, its consent mechanism is relatively "disempowered" because the individual is the last entity consulted in the flow and given little choice, and its consent withdrawal mechanism is relatively inconvenient.

OpenID Connect is effectively a simple OAuth-protected API that does single sign-on and identity data retrieval jobs. Its main innovation is to use lightweight technology to remove friction from tasks that the older SAML standard proved too heavy to tackle in practice.

UMA is innovative because it puts the individual resource owner, and the authorization service that executes their policies for access, at the center of the equation. It enables use cases from proactive delegation ("Share" by user choice) to reactive consent (access approval when asked) to any-time monitoring and adjustment of access (denial and withdrawal), all with a choice to adjust scopes of access at any time. Further, its architecture enables the resource owner to manage these choices in a central location (where "central" is relative to some identity ecosystem that the services used by the individual are willing and able to participate in).

As a final note, the Health Relationship Trust (HEART) standards effort is key because it specifically focuses on patient-centric, privacy-sensitive health data sharing use cases, and it seeks to tighten both the security of the above three standards and their interoperability when applied to the FHIR API.

Implications of UMA for the future of consent as data becomes more accessible

The practice of privacy has, to date, largely and understandably been an exercise in data protection and risk mitigation. However, increasing pressure is coming from several directions:

- Consumer skepticism and cynicism due to press revelations about surveillance, breaches, and bad corporate behavior
- Increasing data volumes and sources due to consumer/health IoT trends
- Worldwide regulatory pressure in new directions, particularly stressing human autonomy and consent
- Businesses striving to demonstrate transparency to consumers in order to build trusted digital relationships with them

The available tools for solving consent problems have been extremely limited to date.

In the web-only era, consent tools consisted largely of "browse-wrap" (individuals were subject to terms of service by merely using a website), notice acknowledgements (such as those compliant with the EU Cookie Directive), and the familiar opt-in and opt-out checkboxes. The only truly interesting consent tool was

the proprietary “Share” button found in Google Docs and similar applications, which acts a bit like online consent directives.

In the API/mobile era, consent tools expanded to include “API-wrap” (client app developers were subject to terms of service appearing in developer documentation by merely calling an API) and OAuth as discussed above.

Finally, now that we find ourselves in an API-inflected “IoT economy,” UMA offers a basis for consent tools that can robustly and strategically address consumer doubt, data volumes and sources, regulatory pressure, and businesses’ need to build trusted digital relationships with users.

Thank you.

/s/

Eve Maler – VP Innovation & Emerging Technology, ForgeRock