Giving users insight into private resource usage through declaration-driven frameworks

Paul van Der Walt, Charles Consel

Giving users insight into private resource usage through declaration-driven frameworks

Paul van der Walt
<paul.vanderwalt@inria.fr>

Charles Consel
<charles.consel@inria.fr>

Phoenix research group, INRIA Bordeaux, France

Empowering users by giving platform providers the technical means to compel developers to provide privacy guarantees.

Context

- Declaration-driven programming frameworks for open platforms.
- Open platforms distinguish platform providers (e.g. Google) from 3rd party developers.
- Declarations allow promises about resource usage and control flow.

Observations

- Declaration-driven frameworks are widespread, but lack expressive declarations.
- Real need for users going unaddressed (Wei 2012).
- In the interest of users, platform providers should require developers to provide guarantees, via declarations.

Goals

- Insight into the usage of private resources (hardware as well as data)
- Providing guarantees about the use of resources
- Demonstration in disparate languages

Resource usage corresponds to visibility in a security context.

Compile-time and run-time verifications in our approach ensure that apps conform to declarations.

Claim

Guarantees can be provided in a language-agnostic way: in functional *or* object-oriented languages, with strong types systems *or* dynamically typed.

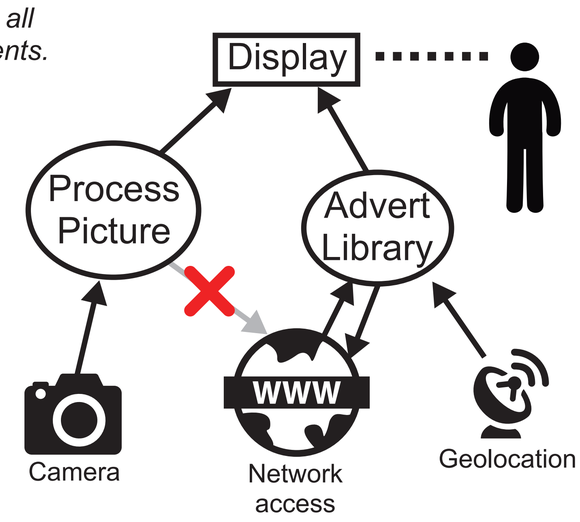
→ Coupled with the benefits of expressive declarations, this must encourage adoption

Example

Real-life scene:

Example: smartphone app, free, ad-supported, takes picture and applies some fancy filter. The advertisement is customised based on your geolocation.

Schematic view of declarations. Note that not all resources are visible to all components.



In a standard Android application, an app with access to camera and Internet might exfiltrate your photos, as well as anything else the app has access to (blocked in our system).

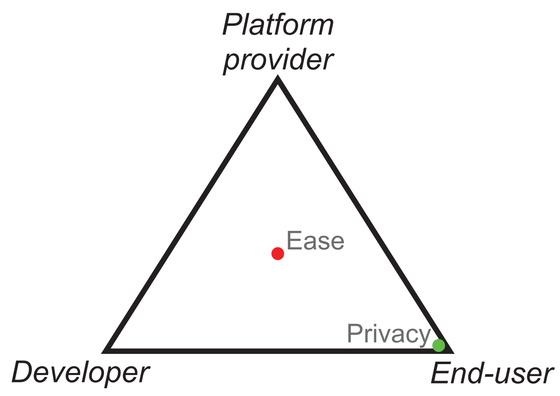
Achievements

Resource usage is elucidated and enforced, as opposed to approaches where the user only sees permission lists (e.g. Android). This gives the user insight into what will happen with their private data.

Users should demand this transparency. It is not in platform providers' best interests (e.g. advertisement revenue increases if profiling becomes more accurate).

Tailoring each component's programming interface also makes development easier for the 3rd party developer. This is favourable for all stakeholders. The user has a larger choice of applications, the platform provider sees increased adoption, and the development requires less effort.

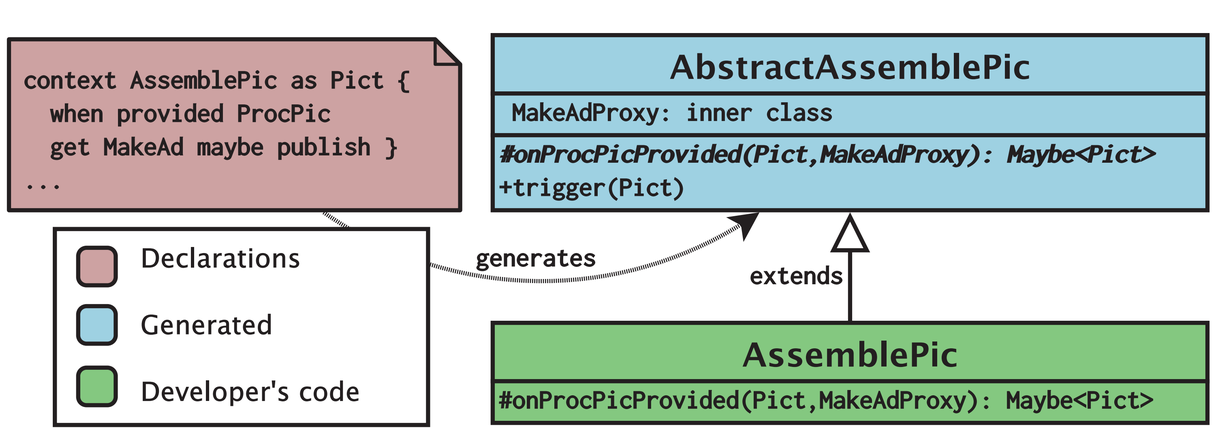
Stakeholder interests



Prototype implementation approaches

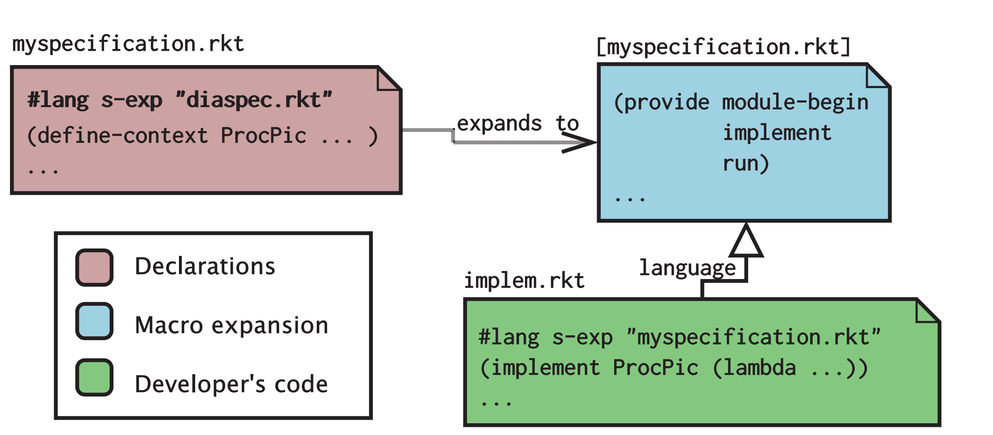
Java – statically typed, object-oriented

Specifications are compiled into an application-specific framework, which uses the Java type system as well as run-time verification to ensure the declared properties are adhered to.



Racket – dynamically typed, functional

Specifications are written using a specially crafted DSL. These expand to another, tailored, language, with which a developer can provide implementations for the various declared components.



References

van der Walt, Consel, Balland, "Declaration-driven frameworks: a language-agnostic approach", manuscript, 2014. <http://phoenix.inria.fr/paul-van-der-walt>
Cassou, Damien, et al. "Toward a tool-based development methodology for pervasive computing applications." Software Engineering, IEEE Transactions on 38.6 (2012): 1445-1463.
Wei, Xuetao, et al. "Permission evolution in the android ecosystem." Proceedings of the 28th Annual Computer Security Applications Conference. ACM, 2012.
Tobin-Hochstadt, Sam, et al. "Languages as libraries." ACM SIGPLAN Notices. Vol. 46. No. 6. ACM, 2011.
Mann, Christopher, and Artem Starostin. "A framework for static detection of privacy leaks in android applications." Proceedings of the 27th ACM Symposium on Applied Computing. ACM, 2012.

Inria INVENTEURS DU MONDE NUMÉRIQUE
UNIVERSITÉ DE BORDEAUX