Erebus: Access Control for Augmented Reality Systems

Sanket Goutam*, Yoonsang Kim*, Amir Rahmati, Arie Kaufman

Stony Brook University

*equal contribution
Two form factors for building AR Systems

**Standalone**
- Oculus Quest 2
- HoloLens 2

**Companion Device**
- Toshiba dynaEdge
- Rokid Air Pro
Applications derive information from device sensors.
How are these applications developed?

Augmented Reality Applications

Development Frameworks
- UnReal
- Unity
- Android

AR SDK libraries
- ARCore / ARKit
- Third party SDK

Target platform
- Android / iOS / Windows
- Wearable Unit / Mobile Phone / PC
Dichotomy between data required and access requested.

Developers use high-level APIs to access sensor data.

Permission enforcement applied only on the target platform.
### Permission Control similar to Smartphone OS.

<table>
<thead>
<tr>
<th>AR Device Type</th>
<th>Device Name</th>
<th>Platform</th>
<th>Access Control Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Standalone Wearable</strong></td>
<td>Meta Quest 2 [43]</td>
<td>Android</td>
<td>App Manifest</td>
</tr>
<tr>
<td></td>
<td>Microsoft HoloLens 2 [44]</td>
<td>Windows</td>
<td>App Manifest, Policy CSP</td>
</tr>
<tr>
<td></td>
<td>Magic Leap 2 [16]</td>
<td>Android</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Google Glass Enterprise [23]</td>
<td>Android</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ThirdEye X2 MR Smart Glasses [22]</td>
<td>Android</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Vuzix Blade AR [70]</td>
<td>Android</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Snap Spectacles [67]</td>
<td>Android</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Raptor AR Headset [19]</td>
<td>Android</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kopin Solos [36]</td>
<td>Android</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Xiaomi Smart Glasses [68]</td>
<td>Android</td>
<td></td>
</tr>
<tr>
<td><strong>With a Companion Device</strong></td>
<td>Lenovo ThinkReality A3 [40]</td>
<td>Android</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Epson Moverio [18]</td>
<td>Android</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Toshiba dynaEdge [63]</td>
<td>Windows</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rokid Air Pro [52]</td>
<td>Android, iOS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NReal Light [46]</td>
<td>Android</td>
<td>No AC mechanism</td>
</tr>
<tr>
<td></td>
<td>Viture One [69]</td>
<td>Android</td>
<td>No information available</td>
</tr>
<tr>
<td></td>
<td>Dream Glass Flow [66]</td>
<td>Android, iOS</td>
<td>No information available</td>
</tr>
</tbody>
</table>

- `<uses-feature android:name="android.hardware.camera" android:required="true" />`
- `<uses-permission android:name="android.permission.record_audio" android:required="true" />`
- `<uses-feature android:name="android.hardware.location.GPS" android:required="true" />`
- `<uses-feature android:name="android.hardware.sensor.heartrate" android:required="true" />`
Developer specifies an access policy to user on Play Store.

*Permissions notice*

Location: used to recommend cultural sites and events based on your current location

Camera: used to recognise artworks and provide related information about them
User installs the app on their device.

**App Manifest**
- Location
- Microphone
- Camera

**AR App**
- Plane Detection
- Face Detection
- Object Detection

**ARCore SDK**

**Prompt user to grant access to sensors.**

**Device Sensors**

**Access sensor data.**

**Grants permissions on first use.**

**Smartify: Arts and Culture**
- About this app
- Developer specified
- App Policy in Android

*Permissions notice*
- Location: used to recommend cultural sites and events based on your current location
- Camera: used to recognise artworks and provide related information about them

**Version**
- 8.1.3

**Updated on**
- Jan 24, 2023

**Requires Android**
- 8.1 and up

**Downloads**
- 500,000+ downloads
Malicious app can violate app policy.

AR Core SDK
- Plane Detection
- Face Detection
- Object Detection

App code

Device Sensors
- Access sensor data.

App Manifest
- Location
- Microphone
- Camera

AR App

Prompt user to grant access to sensors.

Grants permissions on first use.

Smartify: Arts and Culture
About this app
Developer specified App Policy In Android

*Permissions notice*
Location: used to recommend cultural sites and events based on your current location
Camera: used to recognise artworks and provide related information about them

Version
8.1.3
Updated on
Jan 24, 2023
Requires Android
8.1 and up
Downloads
500,000+ downloads
Can we reimagine Access Control for VisionOS?

G1. How can we regulate direct access to sensors?

G2. How to ensure a least privilege access based on developer-specified policy, allowing access to what’s required and nothing more?

G3. Can we allow users to adjust access based on their requirements?
Erebus: regulating sensor access at the OS-level

Augmented Reality Applications

Development Frameworks

- Unreal
- Unity
- Android

Special purpose OS

- ARCore / ARKit
- Ancillary SDKs

VisionOS | Android

Target platform

- Wearable Unit / Mobile Phone / PC

Integrating AR functional requirements with permission control at the OS-level.
Erebus: policy specification language that expresses functionality

- Coarse-grained access requirement. *(Location, Camera)*
- Functional requirement cannot be enforced by the system *(recognize artworks).*

- Functional description in a semi-structured natural language format.
- Fine-grained permission enforcement.
Erebus: users define *what, when, and where* data can be accessed

System validates app's sensor request based on context-dependent policy specification, ensuring *least-privilege*. 
Erebus: preventing sensitive data leakage

Object detection is an imperfect process. False positives could leak sensitive information to the app.
How does **Erebus** prevent leaking sensitive data due to false positives?

We leverage *conflation* technique to optimize object detection accuracy and reduce false positives in Erebus.
Does **Erebus** incur additional latency over API calls?

<table>
<thead>
<tr>
<th>API Type</th>
<th>Erebus (ms)</th>
<th>Unprotected (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera sensor-based API</td>
<td>0.35±0.12</td>
<td>0.18±0.04</td>
</tr>
<tr>
<td>Location sensor-based API</td>
<td>0.22±0.04</td>
<td>&lt;0.01</td>
</tr>
</tbody>
</table>

By enforcing runtime checks on API calls, there is a small overhead incurred by Erebus but this has no impact on performance.
Does Erebus affect app’s overall performance?

<table>
<thead>
<tr>
<th>Component Type</th>
<th>Component</th>
<th>Latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Erebus</td>
<td>Object Detection</td>
<td>28.91</td>
</tr>
<tr>
<td></td>
<td>Non-Max Suppression</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>Object Tracking</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>Conflation</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>Whitelisting</td>
<td>0.08</td>
</tr>
<tr>
<td>Application</td>
<td>Async GPU Readback (Constant)</td>
<td>181.72</td>
</tr>
<tr>
<td></td>
<td>Application Logic</td>
<td>33.47</td>
</tr>
<tr>
<td>Overall Latency</td>
<td></td>
<td>244.46</td>
</tr>
</tbody>
</table>

Our prototype apps were able to run at ~34.16 FPS with Erebus framework enforcing runtime checks.
Erebus: adapting the framework

- Implemented on Google ARCore SDK using Unity Framework.
- Adapted 5 prototype AR applications to our framework.
- We open-source our framework implementation, policy-language design, and prototype applications for developer’s reference.
Erebus: Access Control for Augmented Reality Systems

Sanket Goutam*, Yoonsang Kim*, Amir Rahmati, Arie Kaufman
Stony Brook University

* equal contribution

https://github.com/Ethos-lab/erebus-AR_access_control
https://sgoutam.github.io
@sanketgoutam