Securing recognizers for rich video apps

Chris Thompson, David Wagner
UC Berkeley

How should developers build rich video apps?

Privacy concerns with rich video apps

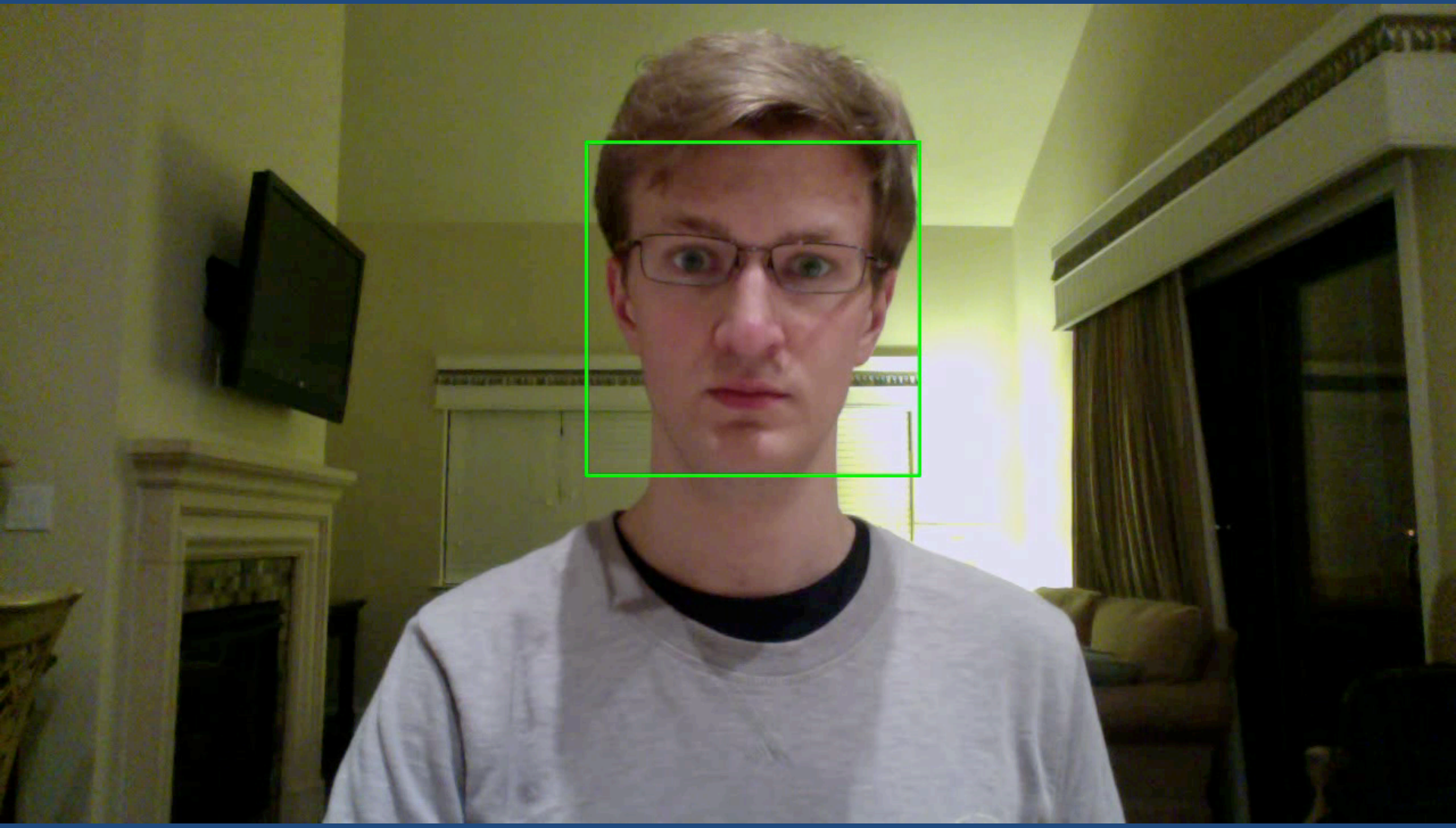
1. Poor use of video by the app
2. Compromise of the app

How can the platform help developers build secure rich video apps?

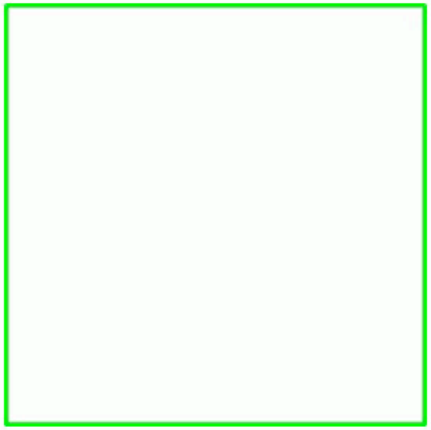
Architecture Goals

- least privilege
- separation of privilege
- privacy by default

Recognizers



Recognizers (revamp)



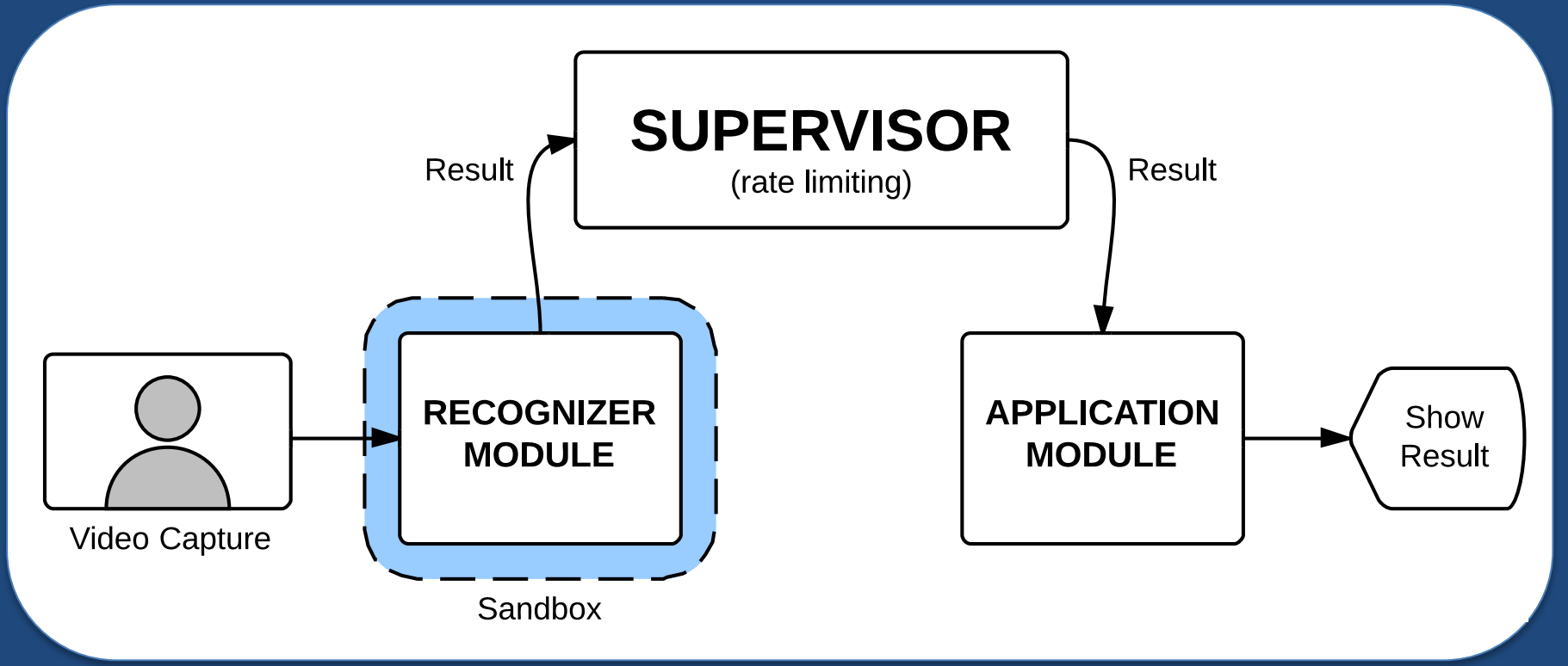
Recognizers (revamp)

```
((300, 200),
  (400, 300))
```

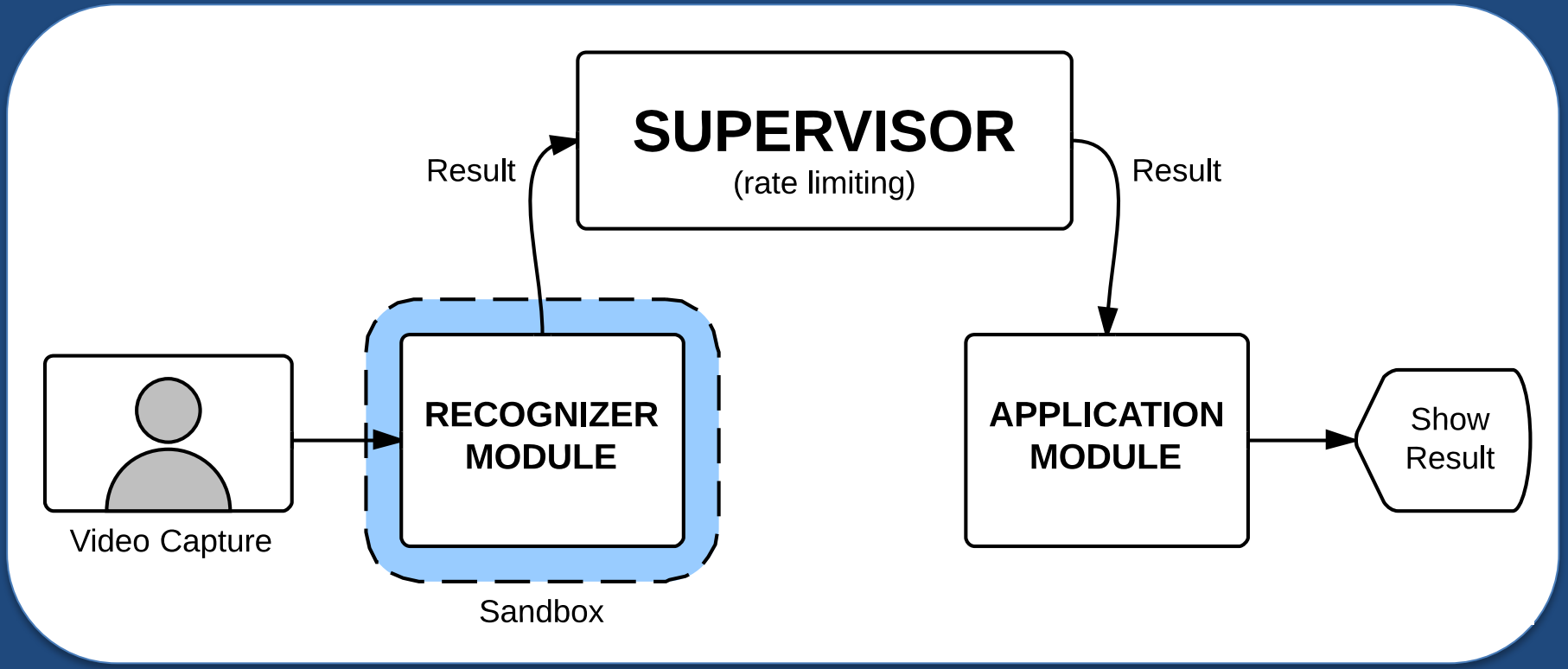
Recognizer

- A function on sensor data stream
  - For this work, we focus on live video
- Extracts useful features/components
  - **Faces**, gestures, objects, locations….
- May maintain some internal state

Design


SUPERVISOR
(rate limiting)
Result
Result
RECOGNIZER MODULE
APPLICATION MODULE
Show Result
Video Capture
Sandbox

Design



Separation of privilege

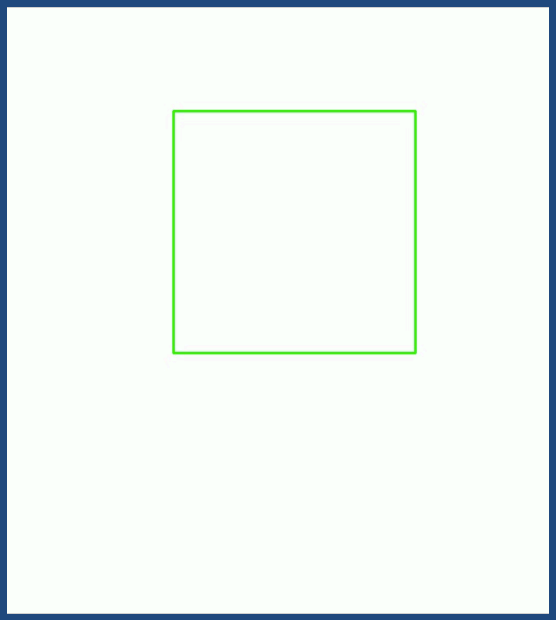
Least privilege

Privacy by default

Sensitive

Embarrassing video

Less Sensitive

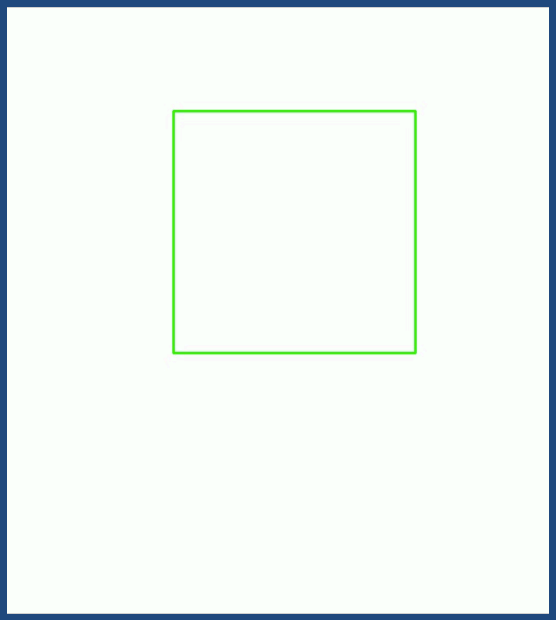


Coordinates of a face in said video

High Bandwidth

Embarrassing video

Low Bandwidth



Coordinates of a face in said video

Implementation

- Python, OpenCV, ZeroMQ
- Separate processes for each module
- Sandbox filesystem and network access
- Bandwidth limits with token bucket

Evaluation

- How hard is it to write apps in this style?
- What's the performance impact?
- How much privacy benefit do we get?

Methodology

- Looked at 17 existing OpenCV applications written in C++
  - Ported into Python, then ported to use our architecture
- Ran each application 80 times on a fixed test video tailored to that application
  - Current results are from a server class machine running Ubuntu

Developer Burden

- Converted 17 computer vision applications written in Python to use our architecture.
  - Simple case: Split main event loop into two, reuse common send/receive code.
  - Sending settings or UI events back to the recognizer module slightly more complicated, but still straightforward.
- Potential for automated or tool-assisted development.

Performance

- Privilege-separated versions are roughly the same as the monolithic versions
  - <10% overhead for 15/17 applications
- Privilege-separation can offer greater opportunities for parallelism and concurrency
  - 11/17 applications saw a speed up of 10% or more

Privacy Benefit

- **Privacy budget**: *how much bandwidth would an attacker need to exfiltrate video with enough fidelity to still recognize faces?*
  - With optimized use of x264 encoding, we found a conservative estimate of **300 bytes per frame**.
- Good apps in our corpus: only extract straightforward features, with small object sizes.
  - 4/17 apps would work as-is
  - 12/17 apps would work if they limited their output to once per second

Security Analysis

- **Can an attacker record embarrassing video and exfiltrate it?** No.
- **Can an attacker extract confidential information from the video and exfiltrate it?**
  - Yes, if the attacker can compromise the recognizer module, but not if they can only compromise the core application module.

Code

- Architecture code and example applications is available on GitHub: github.com/christhompson/recognizers-arch
- Apache 2.0 licensed

Conclusion

- Our security architecture for recognizer applications helps to secure visual recognizer applications while allowing generic and novel computation on video input.
- Our architecture is **practical**, has a **modest performance impact**, requires **little developer burden**, and provides **significant privacy and security benefits**.