Controlling UAVs with Sensor Input Spoofing Attacks

Drew Davidson¹
Hao Wu¹
Robert Jellinek¹
Thomas Ristenpart²
Vikas Singh¹
• Introduce **sensor input spoofing attacks** to exercise an **implicit control channel** over an autonomous vehicle through its **sensors**
• Demonstrate an instance on **optical flow** for two consumer UAVs
• Propose mitigation techniques through **robust algorithms**
Outline

• Introduce **sensor input spoofing attacks** to exercise an **implicit control channel** over an autonomous vehicle through its **sensors**
• Demonstrate an instance on **optical flow** for two consumer UAVs
• Propose mitigation techniques through **robust algorithms**
• Discuss additional defenses and recommendations
Pop Quiz

What is This?

- A tile floor
- An *image of* a tile floor
What Happens if you Fool a Sensor?

• Depends on how sensor is deployed
• Autonomous Vehicles
  – Self-driving cars (Google car)
  – **UAVs (Drones)**
    • Safety critical
    • Commodity sensors
    • Widely used
• Our work:
  – (To our knowledge) first to exercise **continuous control** over UAV motion
Sensor Input Spoofing Attacks

- **Goal:** exercise control over UAV’s actions
- **Adversary:** No physical access to UAV
- **No EMI**
- **Limited Environment Access**
- **Implicit channel**

UAV

Sensor
• Introduce sensor input spoofing attacks to exercise an implicit control channel over an autonomous vehicle through its sensors

• Demonstrate an instance on optical flow for two consumer UAVs

• Propose mitigation techniques through robust algorithms

• Discuss additional defenses and recommendations
Background: Optical Flow (OF)

- **Goal:** quantify motion between two temporally similar images
- **Use in UAVs:** lateral stabilization
  - Sensor: downward-facing camera
    - High framerate
    - Low resolution
- **Sensor detects motion \((x,y)\)**
  - UAV assumes drift \((-x,-y)\)
  - Corrects with motion \((x,y)\)
• **Sparse OF** – only tracking features rather than each pixel

• **Classic:** Shi-Tomasi **corner** detection
  – Sharp intensity falloff along both $x$ and $y$ dimensions
• Produce feature motion vector
  \[ v_1, \ldots, v_n \]
  for each of the N features

• Final motion pair V is component-wise mean of
  \[ v_1, \ldots, v_n \]

\[ V = x, y \]
• Adversary-controlled features

• Move features in the image by \((x,y)\)
  
  – UAV thinks the features are stationary and it is drifting by \((-x,-y)\)
  
  – UAV “corrects” by matching the adversary’s motion \((x,y)\)
Attack: Creating Features

- Project light onto the OF sensor’s plane
  - Scenario 1: portable projector
  - Scenario 2: laser pointer + filter
2 popular UAVs
- ArduCopter – open source control software, popular amongst UAV enthusiasts, primarily for outdoor use
- AR.Drone 2.0 – closed source, popular amongst hobbyists, some use in professional indoor settings

4 real-world environments
- Tile
- Carpet
- Grass
- Concrete
### Attack: Evaluation

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tile</td>
<td>200</td>
<td>Drift</td>
<td>Fail</td>
<td>Control</td>
<td>Drift</td>
<td>Fail</td>
<td>Control</td>
</tr>
<tr>
<td>Carpet</td>
<td>150</td>
<td>Drift</td>
<td>Fail</td>
<td>Control</td>
<td>Drift</td>
<td>Fail</td>
<td>Control</td>
</tr>
<tr>
<td>Concrete</td>
<td>138</td>
<td>Stable</td>
<td>Control</td>
<td>Control</td>
<td>Stable</td>
<td>Control</td>
<td>Control</td>
</tr>
<tr>
<td>Grass</td>
<td>438</td>
<td>Stable</td>
<td>Fail</td>
<td>Fail</td>
<td>Stable</td>
<td>Fail</td>
<td>Fail</td>
</tr>
</tbody>
</table>

- **Portable projector**
  - Only works in low-light at close range
- **Laser pointer**
  - Effective in all but the most feature-rich environments
  - Unbounded motion
  - Rapid enough motion with AR.Drone to cause damage to UAV
Attack: Refinement

- Performed experiments in simulation and practice

- Considered the effect of adversary’s
  - feature light intensity
  - feature pattern
  - feature shape
  - feature size

(full details in the paper)
• Introduce **sensor input spoofing attacks** to exercise an **implicit control channel** over an autonomous vehicle through its **sensors**

• Demonstrate an instance on **optical flow** for two consumer UAVs

• Propose mitigation techniques through **robust algorithms**

• Discuss additional defenses and recommendations
Defenses

- Enhance OF to deal with adversarial features
- Intuition: address the algorithmic limitations of sparse-LK in OF
Random Sample Consensus: RANSAC

- Assume data contains correct “inliers” and bad “outliers”
- Randomly sample k features, each with a “motion hypothesis”
  - Other features vote for each hypothesis based if their own motion is close
- Use the winning hypothesis

Works when adversary lacks majority of features
Breaks down when the adversary overwhelms benign features
Weighted RANSAC w/ Momentum

- **Goal:** assign more weight to trusted features
  - Features accrue weight
  - Fits the scenario of attacker entering scene
- **Smaller number of trusted features can still form correct hypothesis**
Defense Evaluation: Methodology

- Evaluation via simulation
  - Add moving grid of laser “dots” across real image frames
- Several environments
  - Asphalt
  - Carpet
  - Grass
- Used the strongest adversary from our attack strategy
• Tested three variants:
  – Lucas-Kanade (avg): blue
  – RANSAC: red
  – Weighted RANSAC: teal
• LK moves reliably
• RANSAC initially strong until overwhelmed
• WRANSAC fairly steady

![Graph showing motion pixels over frame count for different methods.]
Outline

• Introduce **sensor input spoofing attacks** to exercise an **implicit control channel** over an autonomous vehicle through its **sensors**

• Demonstrate an instance on **optical flow** for two consumer UAVs

• Propose mitigation techniques through **robust algorithms**

• Discuss additional defenses and recommendations
Sensor Firmware Robustness

• RANSAC and Weighted RANSAC are a good first step
  – Likely much better performance to be had

• Key insight: safety-critical sensors need to go beyond random noise
Hardware-level Robustness

- Better cameras mean more features
  - More features complicate the attacker’s goal
- IR illumination + IR cameras for low-light conditions

$I(x + \delta x, y + \delta y, t + \delta t)$
Beyond Robust Sensing

• Consider a stronger adversary
• The “Sombrero Attack”
  – Adversary obscures the entire ground plane
  – Beyond the limits of algorithmic hardening
• Consider *plausible input* requirement
  – Cross-check the results of multiple sensors
  – Drift should be accompanied by acceleration
Future Work: Verifying Sensor Fusion

- Dataflow on firmware
  - Sources: function containing sensor reading
  - Sinks: function containing response
- Policy for desired sensor fusion
- Prototype static analysis on LLVM
Future Work: Considering other SISAs

- Combine SISA with jamming attacks from the literature
- Attack other sensors
Summary

- Introduced Sensor Input Spoofing Attacks on passive sensors
- Crafted attack against Optical Flow on two commercial UAVs
- Developed defenses with robust algorithms
- Recommended future work by hardening the entire sensor pipeline
Thanks

• Questions?

• Page:
  – http://pages.cs.wisc.edu/~davidson/sisa/

• Contact:
  – Drew Davidson
    • davidson@cs.wisc.edu
    • drew@davidson.cool