**Motivation**

**Embedded Systems (ES)**
- Computers that interface with the real world
- Found in day to day life
- 1 car ≈ 60 ES

**Problem: Security**
- ES now connecting to “The Cloud”
- Internet = More Vulnerabilities
- Need for improved systems + security

**Solutions:**
- Virtualization
  - The path to isolation
  - Isolation = increased fault tolerance
- Smaller System Components:
  - Greater compositional flexibility
  - A sleeker, more tailored lightweight system
- Use of Legacy code:
  - No longer need heavy man power
  - Tested code = safer code

---

**Current Virtualization**

<table>
<thead>
<tr>
<th>No Virtualization</th>
<th>Unikernel</th>
<th>Full VM</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="no_virtualization.png" alt="Diagram" /></td>
<td><img src="unikernel.png" alt="Diagram" /></td>
<td><img src="full_vm.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

- **No Virtualization**
  - + lightweight
  - - No isolation
  - - No legacy support
  - - low fault tolerance

- **Unikernel**
  - + Some legacy support
  - - Supports only one application
  - - No isolation

- **Full VM**
  - - Heavyweight
  - - Memory intensive
  - - Expensive I/O Path

---

**Results**

Our new virtualization platform (eVM) allows for:
- Porting of legacy code + Application environment isolation + Carve out unneeded code + COS hypercall layer = a sleeker, more tailored system + a safer system + faster system development

---

**A New Embedded System Virtualization**

**The RUMP Kernel + eVM**

- Uses the Composite OS (COS) as host
- Leverages COS's philosophy of fine grained components
- Runs RK as an isolated user space component
- Ability to pick and choose system services per RK
- COS exports a layer of "hypercalls" to the RK
- Creates an app specific environment
- All possible thanks to COS's fast InterProcess Communication (IPC)

**Introducing the Embedded Virtual Machine (eVM)**

The Rump Kernel (RK)

- + Lightweight
- + Legacy support
- + Isolation
- + Scalable kernel size
- + Multiple Applications

---

**Future Work**

Keep Pushing Granularity

- Application specific protection domains
- RK component specific protection domains
- Treat each ported RK component as a server in user space