A Paravirtualized Android for Next Generation Interactive Automotive Systems

Soham Sinha, Ahmad Golchin, Craig Einstein, Richard West

Department of Computer Science, Boston University
Interactive Automotive Systems
Interactive Automotive Systems
Interactive Automotive Systems + ADAS services

- ADAS Services
  - Lane detection
  - Cruise control
- Control ADAS configurations in real-time
Existing Interactive Automotive Systems

Ford SYNC® 4
TESLA
Toyota | Entune
Existing Interactive Automotive Systems
Advantages of Android in Automotive Systems

- Android Automotive OS
  - Android Auto
- Running Android natively on a vehicle’s interactive system
Limitations of Existing Automotive Systems

No Real-time I/O
Limitations of Existing Automotive Systems

IVI Systems in Volkswagen, Audi Vulnerable to Remote Hacking

by Milena Dimitrova | May 14, 2018 | 0 Comments  audi, car vulnerability, volkswagen, vulnerability

Not guarded enough from security attacks

International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment
DIMVA 2017: Detection of Intrusions and Malware, and Vulnerability Assessment pp 185-206 | Cite this

A Stealth, Selective, Link-Layer Denial-of-Service Attack Against Automotive Networks
Requirements of a Next Generation Interactive Automotive Systems

- Familiar, straightforward and rich user-interface
- Ability to develop custom apps
- Real-time, predictable and secure I/O
- Minimal hardware complexity
Our Solution:

A Paravirtualized Android in Quest-V Hypervisor
Paravirtualized Android in Quest-V Hypervisor
Paravirtualized Android in Quest-V Hypervisor

- Real-time Control Tasks and Sensor Data Processing (e.g., Torque vectoring, Battery management, ADAS)
- Quest Real-Time Kernel (ring 0)
- VMM (ring -1)
- User Apps
- Java API
- Android Runtime and Native Libs
- HAL
- Android Kernel (ring 0)
- VMM (ring -1)
- Secure Shared memory
- Core 1
- Memory
- USB-CAN
- Serial Port
- Core 2
- Memory
- Bluetooth
- WiFi
- Touchscreen

Ring 3

Hardware
Advantages of Paravirtualized Android in Quest-V

- Real-time and predictable I/O in Quest
- Secure I/O data transfer through shared memory between Android and Quest

A single-board solution

- Familiar Android UI
- Large App developer community

Real-time Control Tasks and Sensor Data Processing (e.g., Torque vectoring, Battery management, ADAS)

User Apps
Java API
Android Runtime and Native Libs
HAL

Secure Shared memory

Quest Real-Time Kernel (ring 0)
VMM (ring -1)

Android Kernel (ring 0)
VMM (ring -1)

Core 1
Memory
USB-CAN
Serial Port

Core 2
Memory
Bluetooth
WiFi
Touchscreen
Timing Predictable I/O in Android

- Real-time Quest Functional Services
- Quest Kernel
- Real-time I/O Devices
- Core 1
- Android Services
- Android Kernel
- Non-RT I/O Devices
- Core 2

- Real-time I/O
- Real-time shared memory communication
- Non-Real-time I/O
Promising Preliminary Evaluation - Cold Startup Time

<table>
<thead>
<tr>
<th></th>
<th>Vanilla Android</th>
<th>Paravirtualized Android in Quest-V</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Booting Android</strong></td>
<td>16.6 s</td>
<td>23.7 s</td>
</tr>
<tr>
<td><strong>IVI (HVAC) App Startup</strong></td>
<td>49 s</td>
<td>59.2 s</td>
</tr>
</tbody>
</table>
Timing Predictable I/O in Android

I/O using Paravirtualized Android in **Quest-V**

I/O using vanilla Android
Promising Preliminary Evaluation - I/O Performance

Synchronous USB-CAN I/O Latency

Synchronous USB-CAN I/O Throughput
Future Work

- Supporting Automotive APIs for Quest-V design
- Communication timing requirements
- Secure communication interface
- Power Management
Thank you!
Questions?