CYBERSEC 2024 臺灣資安大會 | 5/14 Tue – 5/16 Thu 臺北南港展覽二館

Generative Future

AIoT & Hardware Security Summit

軟體定義汽車的電控單元資安設計

Static strategy stale? ECU cybersecurity design approach in the SDV era

朱益宏 Ian Chu

研究員

ianyhchu@gmail.com

我是誰

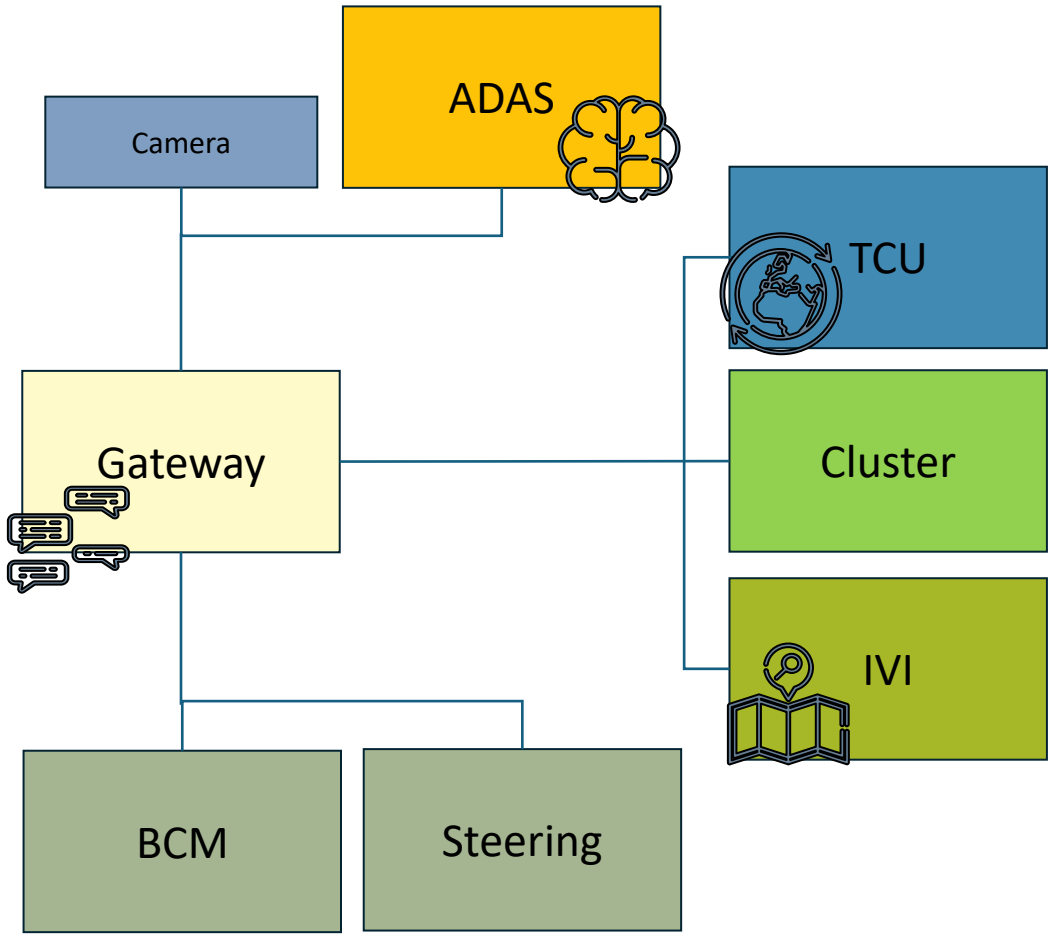
- 歐美車廠SDV資訊安全架構師
- 歐美車廠 TCU系統資訊安全設計師
- 歐美車廠 TCU系統軟體開發
- 中國車廠IVI 系統軟體開發
- 中國車廠OTA 升級系統架構師
- 台灣車廠TCU 系統軟體開發

什麼是SDV (軟體定義汽車)

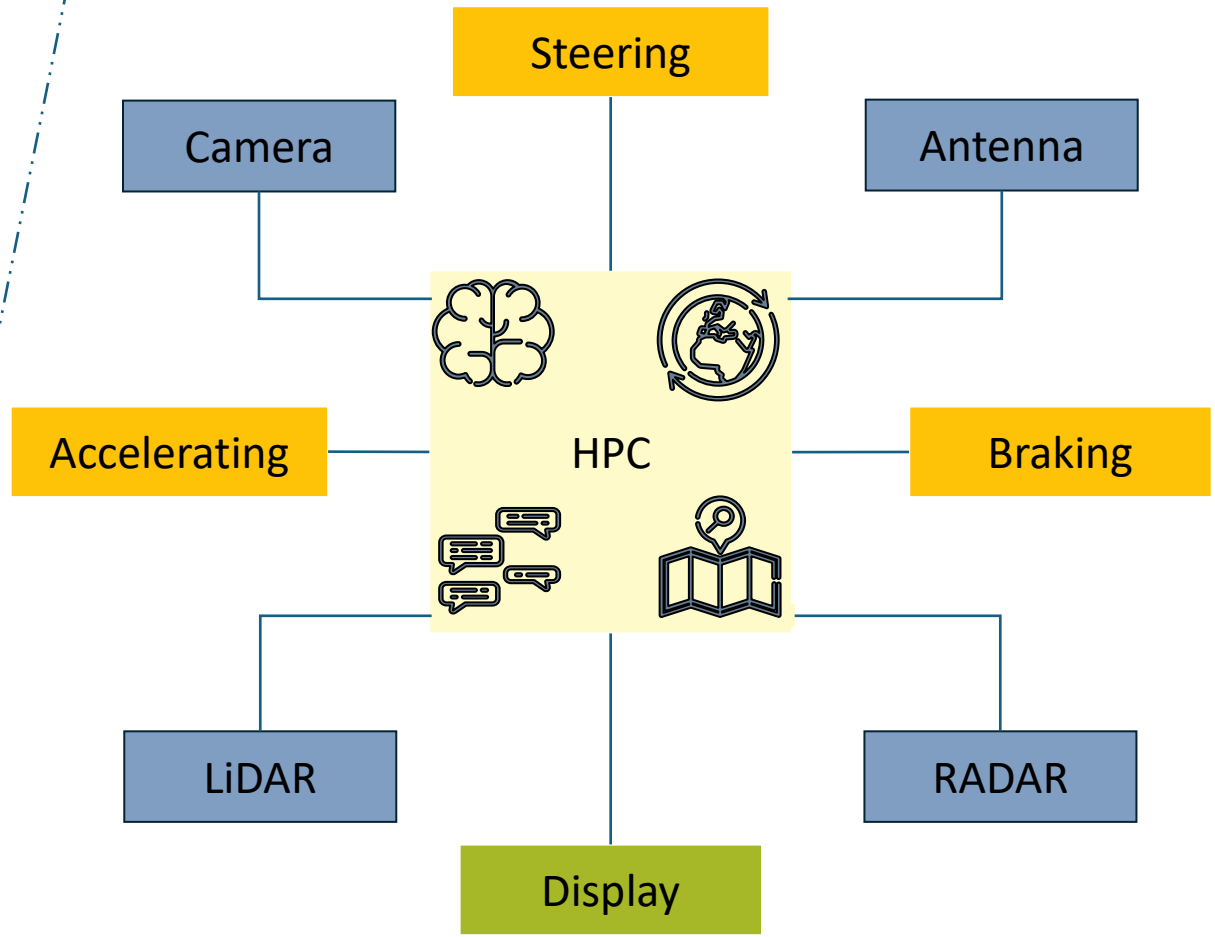
- 透過軟體管理、啟用、新增車輛的功能，提升用車體驗
  - 軟體未至，硬體先行
  - 從落地就折價到花錢買未來
- 連網提供了SDV的基本可行性
  - 遠端管理車輛的能力
  - 軟體疊代的機會
- 整車電子架構(EEA)影響SDV實現的難易度

SDV典型：Tesla Model S: 2012

整車電子架構



Distributed



Centrailzed

汽車將是有輪子的電腦

整車電子與電腦的不同

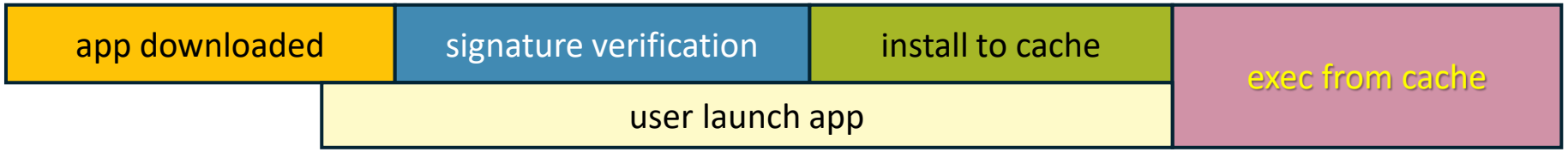
- 設計準則的不同：汽車操作行為是被預期的
  - 主要是為了功能安全 (Functional Safety)
    - ✓ 系統的行為需要經過設計與測試，將非預期的風險降低到可控
  - 也有成本考量：電子元件占整車成本、長期(5年, 10年, …)維運成本
    - ✓ 系統的資源使用率在設計階段安排，盡量最大化資源使用

車用系統的資訊安全考量重點

- 系統完整性 (System Integrity)
  - 確保系統的重要資料(可執行檔、設定)的內容與預期中的一致
  - 更進一步可以支援鑑權(Authentication)，確保系統來自可信來源
- 存取控制 (Access Control)
  - 確保存取系統的重要資料的實體與預期中的一致
  - 最小權限原則減少攻擊面
  - 存取控制設定也是系統完整性要保護的重要資料

系統完整性 (System Integrity)

- 實現方式
  - 安全啟動 (Secure Boot)
    - ✓ RoT (Root of Trust)
    - ✓ ARM PSA (Platform Security Architecture)
  - 程式碼簽章 (Code signing)
    - ✓ Software Package Signing (e.g. APK signature, PGP signature)
- [Note] 避免TOCTOU (Time-of-check to time-of-use)的風險
  - e.g.



存取控制 (Access Control)

- 實現方式
  - 物理介面 (測點、引腳、UART、SPI、USB、JTAG)
  - DAC/MAC
  - 相關網路設定 (防火牆規則，路由表，vlan，apn)
- [Note] 不要依賴登入密碼做為唯一的存取授權機制
  - 難以定期更換密碼
  - 若密碼是唯一機制，應使每一台設備使用不同的密碼，控制損害範圍

動態資料的處理

- 為了系統運作必須寫入的資料
  - 使用者設定、機敏紀錄、資料庫...
  - 仍要確保完整性與鑑權能力，可以使用FDE/FBE (Full Disk Encryption/File-Based Encryption)
    - ✓ 密鑰可以保護在ARM TrustZone TEE或TPM, HSM, SE等硬體中
    - ✓ Android系統透過vold, keymaster TA, kernel keyring 組成密鑰使用流程
    - ✓ Linux系統需要透過PKCS#11介面串systemd, LKUS或dm-crypt類似的流程

SDV所帶來的衝擊

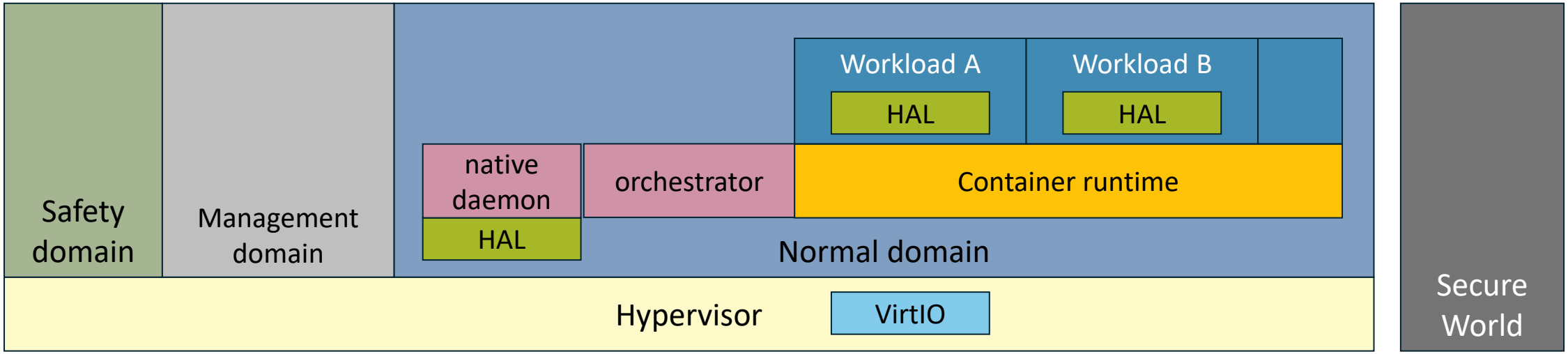
- 透過軟體管理、啟用、新增車輛的功能
  - 軟硬體解耦，軟體分層、模組化支撐快速軟體疊代
    - ✓ Android的各種abstraction layer是很好的例子
    - ✓ Linux的Application Framework則較為分歧
      - AGL, Legato, TelAF
      - 虛擬化

- 問題：如何兼顧軟體彈性並確保系統完整性？
  - Hint: 參考Google對Android碎片化的努力

SDV的一種系統模組化方案

集中式HPC模組化運行應用的一種方案

- 多域隔離明確，多層軟體模組化，適合動態應用管理
- 硬體資源需求大，底層系統設計複雜
  - ✓ 周邊設備配置、Secure World存取，跨域通訊



Takeaway

- 確保多層軟體獨立管理時的系統完整性
  - 確保VM的安全啟動校驗鏈
  - 注意container的校驗是否存在TOCTOU風險
- 小心設計跨域通訊的存取控制
  - 善用硬體提供的能力，e.g. ARM FF-A, SMMU, MMU, EVITA Full HSM
  - 設計各層軟體間的存取控制機制需要考慮開銷
- 安全源自設計，改善永不止息，沒有一勞永逸端點防護方案
  - ISO 21434 Clause 8: Continuous Cybersecurity Activities