Requirements and Architectures for Secure Vehicles

Prepared By: Zi Yi Chen

Prepared For: Professor Marsha Chechik
CSC2125
University of Toronto
Agenda

- Motivation and Background
- Setting Requirements
- Eliminating Weaknesses
- Reasoning about Security and Composition
- Conclusion
- Questions and Discussion
Motivation

- Do you trust the software in your vehicle?
- Iran landed a US stealth drone through a GPS spoofing attack (speculated)
- Self-driving cars require even more software that are vulnerable to cyberattacks
High-Assurance Cyber Military Systems (HACMS) Project

- Goal: construct complex networked-vehicle software securely

- 3 teams:
  - Air team: builds a software stack for unmanned aerial vehicles (UAVs)
  - Ground team: investigate software for automobiles and ground-based robots
  - Red team: professional penetration testers, can access all software, design, documentation etc.

- To build secure software for the air team:
  - UAVs must incorporate third-party software
  - UAVs could be networked to construct systems of systems
  - Must be able to reason about requirements at various abstraction levels
System Decomposition

**FIGURE 1.** The interplay between requirements and architecture. Whether you consider a statement to be a requirement or design decision depends on the abstraction level on which you focus.
Assumptions

An authorized user has the authority to issue any command to the UAV

- Including commands to destroy it
- No limit on what a legitimate user may choose to do

We can’t limit access to the radio spectrum

- Can always launch DoS attack
- Can’t guarantee reception and execution of commands from authorized users
- Can require UAV to reject commands lacking authorization
- Can specify actions the UAV should take to keep itself safe if DoS attack is detected
Construct the Requirements

▪ Focused on variety of known concrete attacks from Common Attack Pattern Enumeration and Classification list (http://capec.mitre.org)

▪ Steps:
  1. Ensured generic security principles
  2. Created system-level security requirements
  3. Additional requirements are imposed
Eliminate Weaknesses

- Also focused on common software weaknesses that lead to security problems from Common Weakness Enumeration website (http://cwe.mitre.org)
- Some weaknesses depend on system architecture
- Other weaknesses can be eliminated by the programming language
Reasoning about Security and Composition

- $\text{Secure}(A) \land \text{Secure}(B) \Rightarrow \text{Secure}(A \oplus B)$
- $\text{MemSafe}(A) \land \text{MemSafe}(B) \land \text{MemSafe}(C) \Rightarrow \text{MemSafe}(\text{System}(A, B, C))$
- $\text{Lem1}(A) \land \text{Lem2}(\text{Chan}) \land \text{Lem3}(B) \land \text{Lem4}(\text{Attack}) \Rightarrow \text{Secure}(A \oplus B)$
Reasoning about Security and Composition

The UAV executes only unmodified commands from the ground station.

The UAV motors only execute commands that pass decrypt.

Only the ground station can send commands that pass decrypt.

The UAV motors only receive commands from ML: flight_gen_process.impl.

All commands to ML: flight_gen_process.impl pass decrypt before reaching the UAV motors.

Decomposition of the claim with arguments for encryption algorithm strength, correct implementation, and encryption key management

Claims about the other three motors here

All inputs to ML: flight_gen_process.impl are internal sensor data or pass decrypt before reaching the UAV motors.

Claims about additional sensor data here

Commands to UAVSWML.motors_and_radios._avlink pass through decrypt before reaching the UAV motors.

Commands to uart5SR_inst: uart5SR not passing through decrypt can't reach the UAV motors.

Decomposition of the claim related to information sent to or from the UAV through the interrupt service routine controlling the radio link

UAVSWML.sensors.gps_data only receives sensor data.

UAVSWML.sensors.battery_data only receives sensor data.

FIGURE 2. A portion of the automatically generated assurance case tree for an unmanned air vehicle (UAV) for the requirement, “The UAV executes only unmodified commands from the ground station.”
HACMS comprises three 18-month phases

Red team receives a demo vehicle and software at the end of each phase

Phase 1: attacks were possible only through communications links between ground station and UAV

Phase 2: provided root access to a Linux partition that controlled a camera used for vehicle tracking

Phase 3: adding secure geofencing to ensure UAVs avoid certain no-fly zones

Vehicles can withstand attacks from sophisticated attackers with:
  - Careful attention to requirements and system architecture
  - Verified approaches to remove known security weaknesses
Discussion

- What do you think about the assumptions?
- Environment changes?
- Results of the third phase?