Attack of the killer Robot: or How to Break Embedded and Mobile Software

Jon Hagar
invision@ecentral.com
This is More of a Workshop

- A little talking head (with charts)
- Embedded
- Mobile
- Robot
- Attack – Tour (concept)
- Holiday Party
Definitions

Embedded Software Systems

- Interacts with unique hardware/systems to solve specialized problems interacting with and/or controlling the “real world”
  - IT software runs with largely are “generic” hardware
  - User is only dimly aware the device is use has software/computers
- Usually have significant hardware interface issues and concerns
  - Initialization, noise, power-up, power-down, timers, sensors, etc
- Often resource constrained
  - RAM, ROM, stack, power, speed, time, etc
- Typically has restricted to no Human User Interface (smart phones still not a PC)
- Have no way (or only a risky way) to update and/or change the software
- Involve risks, hazards, safety, and/or some specialized domain knowledge and logic/algorithms usually controlling hardware
What does Embedded look like?

- **Examples**
  - Avionics systems: planes, cars, rockets, military,…
  - Telecom: switch, routers, phones, cell devices,…..
  - Transportation: traffic control, railroad, trucking, ….
  - Industrial control: lighting, machines, HVAC, nuclear/pc
  - Medical: pacemaker, dispensers, ………
  - Home and office systems: control, entertainment (TV box)
  - Now includes Smart Devices and mobile Apps
Fundamental Software Capabilities

• Whittaker lists 4
  – Software accepts inputs from its environment
  – Software produces output and transmits it to its environment
  – Software stores data internally in one or more data structures
  – Software performs computation using input or stored data

• Embedded refines with
  – Function in/with Time
  – Use/control Unique Hardware (as sub of items 1 and 2)
What is an Attack

• Embedded Software System Testing – In part, the process of attempting to demonstrate that a system (hardware, software, and operations) does not meet requirements, functional and nonfunctional objectives

  – Embedded software testing must include “the system”

• Attack - goes after common modes of failure and bugs to show “does not meet”

• We go after our enemy with many approaches
  – Tools
  – Levels
  – Attacks
  – Techniques
  – Etc
An Attack Is

- Based on a common mode of failure seen over and over
  - Maybe seen as negative, but really a positive
  - Goes after the “bugs” that may be in the software
  - Based on or uses classic test techniques and test concepts
    - Lee Copeland’s book
    - Many other good books

- Testers learn these in a domain after years and form a mental model (most testers attack)

- Organize a few embedded attacks
  - Based on literature research of published bugs
  - Be suspicious
Whittaker’s List

• A good starting point for software attacks in general that can be applied to embedded:
  – User Interface Attacks
  – Data and Computation
  – File System Interface
  – Software/OS Interface

• “How to Break Software” lists 23 attacks
Embedded/Mobile Attack Divisions

- Developer implementation level
- Time
- Hardware
- Operating System
- Software
- Data
- UI/GUI
- Combinations
- Apps
Play Time

• We going to do an exercise now

• 2 groups

• Create an attack (more to follow)
  – Keep it simple

• Then a retrospective (we all should do these)
Attack

• Environment – Control Law Problems
  – Big Hint: Conversion and output of the control signals to the real world

• Wheel alignment case put here
How to determine if the attack exposes failures (in control)

- Any of the team conducting this attack needs to understand the level they are attacking and what the indicators include:
  - System failure (crash and burn)
  - Unhandled control (out of control) situations and cases
  - Singularities, math problems, flow of data
  - Performance issues (time)
  - Building up of error terms because numeric representation, accuracy or precession problems
  - Small input or output “fuzz” where the ideal is not met
How to Conduct the Attack

• Obtain the laws and flow (requirements)
  – Travel in a straight line (+/-5%)
  – Move 10 feet
  – Sound alarm before moving
• Define the data ranges
• Design the attack to target the range end points using, for example:
  – Testing at boundaries
  – Random and/or CT data selection
• Fill out the “charter” (next page)
• Run the attack (tell me what to do)
• Analyze the laws for stability and data problems
• Learn and repeat as necessary
Exploratory Test Card (Charter)

Name of Test:

- What to Test (requirements):
  1. 
  2. 
  ... 
- Success Criteria (data ranges):
  1. 
  2. 
- Support items needed:
- Steps:
  - 1. 
  - 2. 
  - 3. 
  - 
  - 
  - 
  - 
  - 
  - 
  - 
  - 
  - 

Results (bugs, observations, lessons learned, positives, issues, concerns, more risks....)
Feedback - Retrospective

• What did you accomplish?

• What did you think of?

• What would you do differently?

• What favors or opposes this attack?
More Attacks (a partial in work list for a book)

- Control – Computation
- Control – A2D & D2A
- Control – Force outputs
- Control – Long duration runs
- Control – Logic
- Developer – Implementation code
- Developer – Path coverage
- Developer – Static code analysis
- Fault tolerance – Missed cases
- Fault tolerance – Software
- Hardware – Communications
- Mobile - Security
- Hardware – Hw-Sw
- Hardware – Internal
- OS-OS
- Software – Data
- Software – Command
- Software – Software/Interfaces
- Time - Interrupts
- Time – Software
- Time – System Sequence
- User Interface – Alarm
- User Interface – Guards
- User Interface
- GUI
- Apps
- Virtual
Thanks (ideas used from)

- James Whittaker (attacks)
- Elisabeth Hendrickson (sims)
- Lee Copeland (techniques)
- Brian Merrick (testing)
- James Bach (exploratory & tours)
- Cem Kaner (test thinking)

- Many teachers
- Generations past and future
- Books, refs and etc
We Are Done

- Mobile, Apps, Embedded pretty hot right now
- They are everywhere
- They are software
- They are buggy

- We need to test them

- More to come