ECE 471 – Embedded Systems
Lecture 18

Vince Weaver

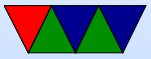
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

8 November 2016

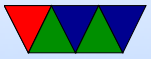
Announcements

- Back from Linux Plumbers
- Working on getting homeworks graded
- Don't forget HW#10, due next Thursday
- Also don't forget project ideas, I'll respond



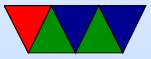
Pi Cluster Paper

- Trouble when trying to evaluate a board for use
- Lots of parameters
  FLOPS
  STREAM
  Watts
  GFLOPS/W
  Integer
- The ones on paper aren't always best. Recent Orange

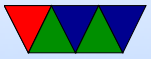


Pi release.

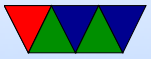
- Really networking is the holdup.
- Nice to have a system to run tests on that is low power. That many x86 machines noisy and power-hungry and hot
- Would probably get a Pi3 if had to do it again



Case Studies

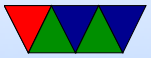


Social Engineering



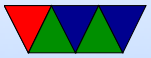
Worrisome embedded systems

- Backdoors in routers.
- Voting Machines, ATMs
- pacemakers
- Rooting phones
- Rooting video games
- Others?



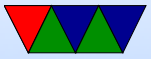
Voting Machines

- Maine has paper ballot — not too bad
- Often are old and not tested well (Windows XP, only used once a year)
- USB ports and such exposed, private physical access
- Can you trust the software? What if notices it is Election Day and only then flips 1/10th the vote from Party A to Party B. Would anyone notice? What if you have source code?
- What if the OS does it. What if Windows had code that



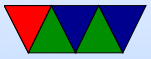
on Election Day looked for a radio button for Party A and silently changed it to Party B when pressed?

- OK you have and audit the source code. What about the compiler? (Reflections on Trusting Trust). What about the compiler that compiled the compiler?
- And of course the hardware, but that's slightly harder to implement but a lot harder to audit.



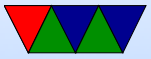
Examples – CANbus

- 2010 IEEE Symposium on Security and Privacy. *Experimental Security Analysis of a Modern Automobile* U of Washington and UCSD.
- Fuzzing/ARM/CANbus
- can control brakes (on / off suddenly)
- heating, cooling, lights, instrument panel
- windows/locks Why? fewer wires if on a bus then



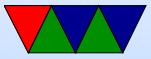
direct-wired

- electronic stability control, antilock, need info from each wheel
- roll stability control (affect braking, turning to avoid rollover)
- cruise control
- pre-crash detection (tighten seatbelts, charge brakes)
- while it might be nice to have separate busses for



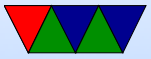
important and unimportant, in practice they are bridged

- Locks– monitor buttons, also remote keyfob... but also disengage if airbag deploys
- OnStar – remotely monitor car, even remotely stop it (in case of theft) over wireless modem
- Access? OBD-II port, also wireless
- 2009 car
- cars after 2008 required to have canbus?

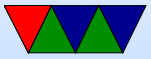


- Problems with CAN

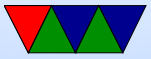
  - Broadcast... any device can send packets to any other
  - Priority.. devices set own priority, can monopolize bus
  - No authentication... any device can control any other
  - Challenge-response. Cars are supposed to block attempts to re-flash or enter debug mode without auth. But, mostly 16-bits, and required to allow a try every 10s, so can brute force in a week.
  - If you can re-flash firmware you can control even w/o ongoing access



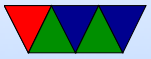
- Not supposed to disable CAN or reflash firmware while car moving, but on the cars tested they could.
- Probing – packet sniffing, fuzzing (easier as packet sizes small)
- experiments – on jackstands or closed course
- controlled radio – display, sounds, chimes
- Instrument panel – set arbitrary speed, rpm, fuel, odometer, etc



- Body control – could lock/unlock (jam by holding down lock), pop trunk, blow horn, wipers on, lights off
- Engine... mess with timing. forge "airbag deployed" to stop engine
- Brakes.. managed to lock brakes so bad even reboot and battery removal not fix, had to fuzz to find antidote
- can over-ride started switch. wired-or
- test on airport. cord to yank laptop out of ODB-II

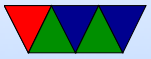


- fancy attacks. Have speedometer read too high. Disable lights. "self-destruct" w countdown on dash, horn beeping as got closer, then engine disable.

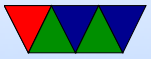


Stuxnet

- SCADA – supervisory control and data acquisition
- industrial control system
- STUXNET.. targets windows machines, but only activates if Siemens SCADA software installed. four zero-day vulnerabilities
  USB flash drives
  signed with stolen certificates

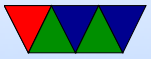


- Interesting as this was a professional job. Possibly by US/Israel targeting very specific range of centrifuges reportedly used by Iran nuclear program. While reporting "everything OK" the software then spun fast then slow enough to ruin equipment.



Examples – JTag/hard-disk

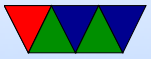
- JTAG/Hard-disk takeover
- `http://spritesmods.com/?art=hddhack&page=8`
- Find JTAG
- 3 cores on hard-disk board, all ARM. One unused.
- Install custom Linux on third core. Then have it do things like intercept reads and change data that is read.



Places for More Info

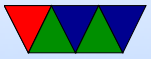
- Embedded projects: `http://hackaday.com`
  They had a recent series on CAN-bus

- Computer Risks and Security Issues: The RISKS digest from comp.risks
  `http://www.risks.org`



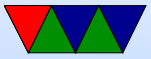
Software Bugs

- Not all bugs are security issues
- Coding bugs can have disastrous effects



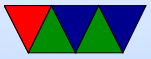
Automotive

- Until recently no standard
- Bugs, Toyota firmware
- `http://www.edn.com/design/automotive/4423428/2/Toyota-s-killer-firmware--Bad-design-and-its-`

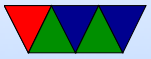


Airplanes

- DO-178B / DO-178C
- Software Considerations in Airborne Systems and Equipment Certification
  - Catastrophic: fatalities, loss of plane
  - Hazardous: negative safety, serious/fatal injuries
  - Major: reduce safety, inconvenience or minor injuries
  - Minor: slightly reduce safety, mild inconvenience
  - No Effect: no safety or workload impact

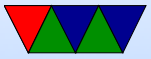


- AA Flight 965. Autopilot to waypoint R. Re-entered it, two starting with R, so it helpfully picked one with highest frequency, did a semi-circle turn to east right into a mountain.

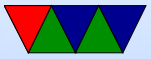


Military

- Patriot missile – clock drift slightly, but when on for hundreds of hours enough to affect missile tracking
- Yorktown smart ship – 1997 – Running Windows NT. Someone entered 0 in a field, divide by 0 error, crashed the ship. Database crash, crashed propulsion system. Rumors that it needed to be towed in, but no, only down for 2.75 hours.
- F-22s computers crashed when crossing 180 degrees



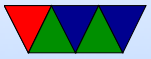
longitude? Lost navigation and communication, had to follow tankers back to Hawaii.



Spacecraft

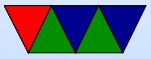
- Mariner 1 (1962) – rocket off course due to mis-transcribed specification into FORTRAN, missing overbar

- Apollo 11 (1969) – landing on moon. Processor normally loaded with 85% load. DELTAH program run which take 10%. But buggy radar device was stealing 13% even though in standby mode. Mini real-time OS with priority killed low-priority tasks so things still worked.

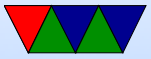


- Ariane 5 Flight 501 (1996) – famous. $370 million. Old code from Ariane 4. Could not trigger on Ariane 4. Horizontal acceleration crashed primary and secondary computers, sending debug messages that the autopilot read as velocity data. Conversion from 64-bit float to 16-bit signed int overflow. Should have had check on it that vertical acceleration, but did not. Not properly simulated in advance. Written in ADA

- NASA Mars Polar Lander (1999) – mistook turbulence vibrations for landing and shut off engine 40m above surface



- NASA Mars Climate Orbiter – ground software using lbf (pound/foot) units, craft expecting Newtons
- NASA Mars Spirit rover (2004) – temporarily disabled due to too many files on flash drive. Constantly rebooting. Radio could understand some commands directly, could reboot with flash disabled. Fixed when deleted some unneeded files. Eventually reformat.
- ESA CryoSat-1 (2005) – lost due to missing shutdown command

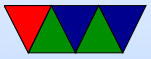


Medical

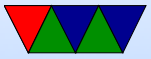
- IEC 62304

- Therac-25 radiation treatment machine. 6 accidents, patients given 100x dose. High power beam activated w/o spreader too.
  Older machines had hardware interlock, this one in software. Race condition. If 8-bit counter overflow just as entering manual over-ride, it would happen.

  – Software not independently reviewed

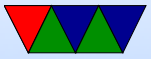


- Something wrong: Printed “MALFUNCTION” and error number 1 to 64 which was not documented in manual. Press P to clear.
- Operators not believe complaints from patients.
- To trigger, had to press X (mistake), up (to correct), E (to set proper) then ”Enter” all within 8 seconds.
- This missed during testing as it took a while for operators to get used to using machines enough to type that fast.
- Used increment rather than move to set flag.
- Written in Assembly Language



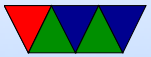
Financial

- Knight Capital. Upgrade 7 of 8 machines, missed last. Re-used a flag definition with new software. Caused massive selloff, $440 million

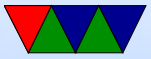


Power

- 2003 Blackout. Race condition, no alarms notify as wires fail, backup of alarms crash server.

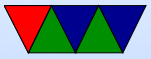


Good Design Practices

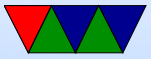


Space Shuttle Design

- HAL/S high-order assembly language (high-level language similar to PL/I)
- PASS software – runs tasks. Too big to fit in memory at once
- BFS – backup flight software. Bare minimum to takeoff, stay in orbit, safely land, fits in memory, monitors pASS during takeoff/landing Written by completely different team.

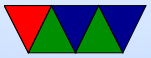


- 28 months to develop new version
- IBM
- originally 424k of core each
- Extensive verification. One internal pass, one external
- 4 computers running PASS, one running BFS
- Single failure mission can continue; still land with two failures
- 4 computers in lock-step, vote, defective one kicked out



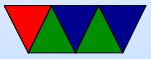
Code Safety Standards

- Avionics: DO-178C (1992 for B)
- Industrial: IEC 61508 (1998)
- Railway: CENELEC EN 50128 (2001)
- Nuclear: IEC 61513 (2001)
- Medical: IEC 62304 (2006)
- Automotive: ISO 26262 (2011)



Writing Good (Embedded) C Code

- Various books. Common one: MISRA: Guidelines for the Use of the C Language in Critical Systems
- Comment your code!
- Strict, common code formatting (indentation)
- More exact variable types (int32_t not int) Size can vary on machine, and on operating system
- Subset to avoid undefined behavior



- Tool that enforces the coding standards
- Good to write safe code even if it isn't meant for a safe application. Why? Good practice. Also who knows who or when your code might be copied into another project.