ECE 471 – Embedded Systems
Lecture 24

Vince Weaver

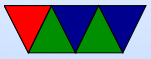
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

4 November 2024

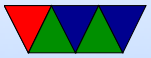
Announcements

- HW#8 was posted, pick up hardware if you haven't
- Don't forget project ideas for Friday



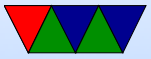
Current Events: Daylight Savings

- Clock math always hard for computers
- Can you hardcode start/stop days for Daylight Savings?
- Congress can (and has) changed it on fairly short notice
- Other countries this happens yearly
- Can help to have system clock UTC + offset rather than trying to have it set to local time



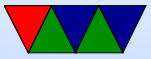
Embedded System Security: Voting

- Election Tuesday
- Are voting machines embedded systems?
- Can elections be hacked?
- Most reports of hacking are social engineering type (i.e. Foreign Country buys deceptive Facebook ads)

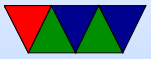


Voting Machine Types

- Vary a lot by state and even town (good is some ways, hard to target them all)
- Old fashioned: purely paper, machines with levers
- Like Maine: mark paper ballot, use embedded system to count, have paper trail if need recount
- Touch-screen voting (often encouraged for accessibility), OK if it prints paper ballot you can later hand count
- Fully electronic with no paper trail, difficult to audit
- Even worse: all internet voting

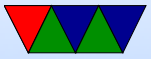


- Mail in votes have their own issues, make votes possibly non-anoymous so people can pressure/bribe you to vote a certain way



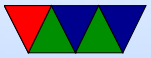
Voting Machine Hacking

- Some voting machines found trivial to hack. Running windows, with exposed USB connector.
- How did researchers get access to them. (eBay)
- Attacks often have to be local unless you happen to hack main database



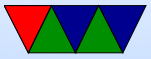
Why hard to audit/secure?

- Often are old and not tested well (Windows XP, only used once a year)
- Run by small teams in towns w/o much IT experience
- Often bought from lowest bidder
- 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



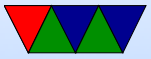
on Election Day looked for a radio button for Party A and silently changed it to Party B when pressed?

- 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.



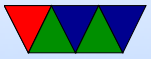
Aside about Shared vs Static Libraries

- Shared libraries, only need one copy of code on disk and in memory
  - Good for embedded system (less room needed)
  - Good for security updates (only need to update lib, not every program using it
- Static libraries, all libraries included
  - No dependencies
- These days maybe containers, docker, kubertenes
- Also Flatpack, Snap. Why? Stability, Know package



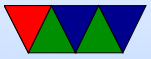
will work on all distributions, Not have to install dependencies

- Can use `ldd` to view library usage



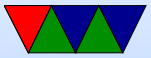
More on Firmware

- Devices are their own embedded systems these days. Lots of small things have microcontroller inside
- What is firmware?
- Code that runs on an embedded system
  - Is it only bare metal?
  - Could a full Operating System be included?
- Traditionally is a binary "blob"
- In ROM? Or upgradable? Why might you want to upgrade? (bug fixes, economy, etc.)



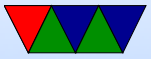
Binary Blobs

- What is in the firmware?
  - Can you modify it yourself?
- Can it contain a full operating system?
  - ThreadX on Pi, Minix on Intel servers?
- Where does it live?
  - Hardcoded in ROM?
  - Upgradable by flash?
  - RAM that's uploaded at boot?



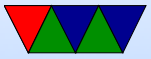
Non-persistent Firmware

- RAM that gets uploaded after boot (loses state on power off)
  What happens if your hard-drive or some other boot critical hardware uses this?
- Can be annoying if writing low-level code, you bought hardware but it might not work at all unless you upload a bunch of code to it each time you power up



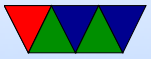
Open Source Firmware Issues

- Can you run a "completely" free computer with completely open software?
- Note the different uses of Free
  - Free as in Freedom/Liberty to use code as you wish
  - Free as in no cost to you
- Depends on how you classify firmware



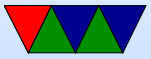
Linux Firmware Issues

- licensing issues
- can boot media (CD, USB) ship with firmware
- What if distro (like Debian) has rules against non-open binaries
- Can you netboot if the network card requires proprietary firmware (common problem with wifi cards)
- Weird workarounds, webcam that had MacOS driver so had to extract firmware from that and copy to Linux partition



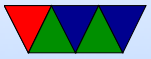
Firmware Licensing Issues

- Is a computer system truly free software if binary blobs running (like on a Pi)
  - Aside, FSF has weird definition where they only get upset about updatable firmware
- Debian tried to have a firmware-free install by default, but had to give up as not practical (especially at install)
- If someone ships firmware that has GPL code in it, what are their responsibilities?

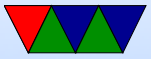


Offloading Work to Firmware

- Companies don't like sharing their code, which makes things like Linux drivers hard
- Can they put as much as possible into firmware with only a small stub in OS?
  - Benefit: OS does less work, is simpler
  - Downside: you have no idea what this secret chunk of code is doing
- On Linux, NVIDIA drivers famously contentious. Possibly they are now planning to push more and more

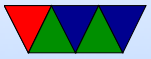


code onto firmware on the card itself



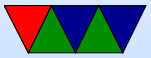
Trusted Firmware

- Firmware can be dangerous: runs below/outside of the Operating System, doesn't matter how secure your OS is if firmware compromised
- Can you trust your firmware to be not-evil?
- Evil Maid problem – what if someone breaks into your hotel room and replaces your firmware – could you tell?

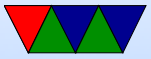


Firmware Dangers

- What if firmware reprogrammed on USB key to be evil?
- What if firmware on USB keyboard is evil? How would you know?
- What if hard-drive firmware reprogrammed
- Do you audit the firmware on your system?
- What could evil firmware do?
  - USB keyboard, log keypresses, send key commands at 2am to open web browser and cut-and-paste data to evil website

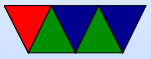


- Disk could refuse to write certain values, or maybe just notice writing a spreadsheet and randomize numbers, etc
- USB key could look like 4GB storage but then later change its ID to a network card and send data



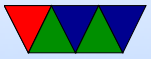
What can we do?

- Epoxy shut USB ports on computers?
- Some secure sites actually do that
- It's a real threat, viruses have spread over USB
  Sneaky people can drop USB keys in parking lots in hopes they'll be plugged in to see who owns it



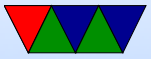
Question

- Instead of epoxying shut USB, just have OS ignore any USB ports?
  - In theory you could do that. Which is cheaper, unskilled labor epoxying USB or else writing your own kernel (or convincing Microsoft) to disable USB?
  - According to spec if OS is ignoring USB, are you safe?
  - (Ignoring the possibility of a USB device with super-capacitors on board to explode your system)
  - Past DMA attacks with Firewire which could bypass



OS and read/write memory

- What if hardware vendor for whatever reason talks to the USB device even though OS hasn't asked it to? It probably shouldn't, but how can you audit that? What about the firmware on the USB controller itself?
- The USB controller hooks up to a main CPU bus somehow, maybe over PCIe, maybe directly to CPU. It can potentially do lots of sneaky things.
- Why would there even be a controller for USB? Well when you plug in has to negotiate power, speed, and other stuff too. Probably a microcontroller



- It all depends how paranoid you want to be
- Even things like sandboxes and if you have all the code, can you *prove* the code is correct? That the compiler is trustworthy? This is actually a field of study, trying to prove code is correct. It's not trivial.
- Doesn't matter how good your software is if SW/HW compromised
- Can you analyze all 10 billion transistors in modern CPU?
- USB keyboard needs to work, even w/o OS