# Embedded Systems Security

Jim Gettys
November 5, 2014

# Pointers

- The Internet of Things is Wildly Insecure...
  Bruce Schneier, Wired, January 2014
- Heartbleed as Metaphor
  Dan Geer, Lawfare Blog, May, 2014
- TAO Catalog
  NSA Ant Division, 2008, Published December 30, 2013
- Familiarity Breeds Contempt: ...
  Sandy Clark, Stefan Frei, Matt Blaze, Jonathan Smith, ACSAC '10
- The Nightmare on Connected Home Street
  Matt Honan, Wired, June 2014

# Software Lifecycle

- We are used to throwing computers away
  - Your phone, laptop, desktops, etc.
  - We've learned through great pain that we **must** keep them updated
- But we now build long lived devices and systems with computers inside, that are Internet connected
  - Your thermostats, home theater, home router, home theater, security cameras, light bulbs, etc. Soon car, refrigerators, coffee makers...
  - Installation costs often greatly exceed cost of the computer
- Some devices have potential lifetimes measured in decades
  - *Timescales are long relative to human organizations*
  - We've presumed we can “forget about them”
  - Is this safe? NO! The SCADA problem writ large

# Familiarity Breeds Contempt:
## The Honeymoon Effect and the Role of Legacy Code in Zero-day Vulnerabilities

By Sandy Clark, Stefan Frei, Matt Blaze, Jonathan Smith, ACSAC '10

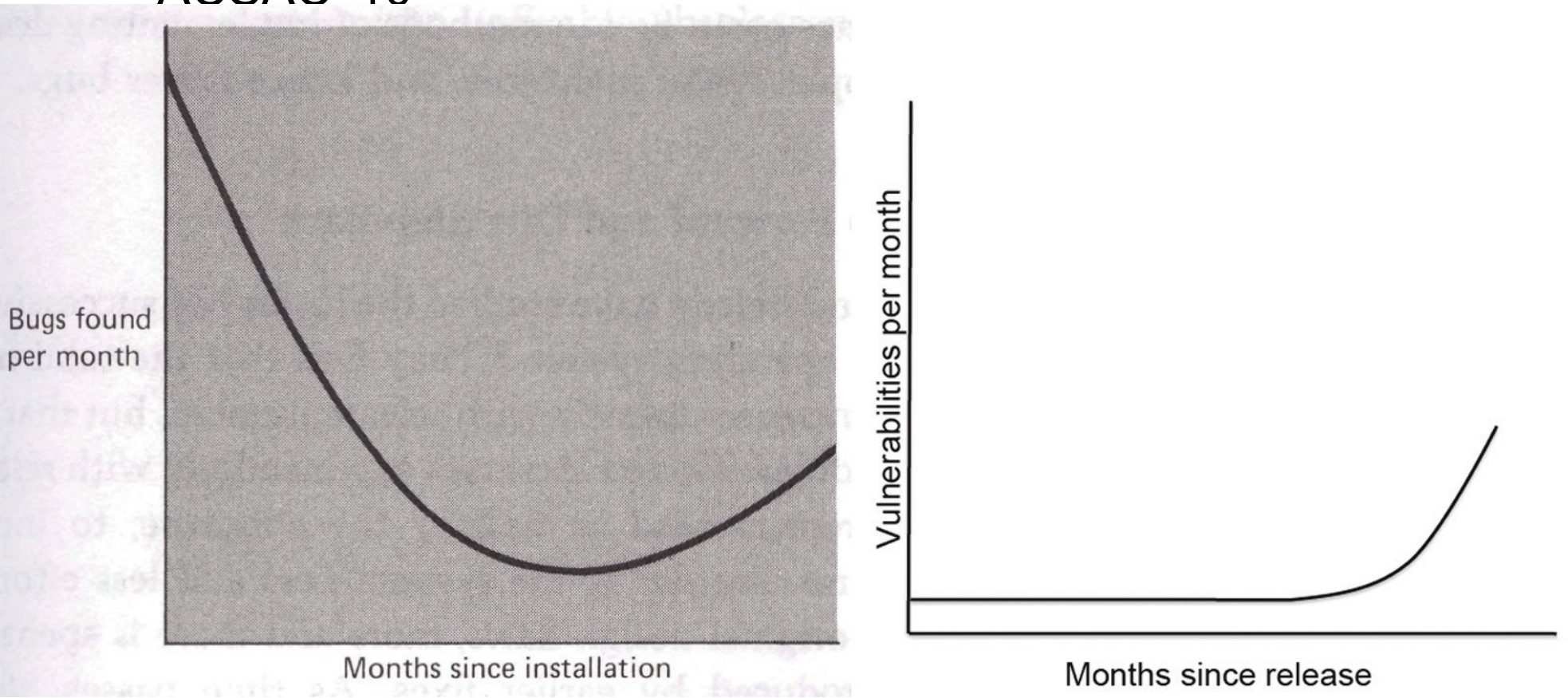


Figure 1

# Device Lifetime is a Cruel Master
# Honeymoon effect take-away

- You cannot leave software and devices “unmaintained”: continuous update is essential, *for the life of the device*
- Products (at least complex ones) **MUST** have **SECURE** update stream for the life of the device!  (Remember Windows XP!)
  - You **must** select components that **CAN** be maintained
  - You **must** select products that **CAN** be maintained
- The owner **must** have ultimate control! You must have ultimate power when the device/network/system fails.....
  - How long will a device remain in the ecosystem? Your router? Your thermostat? Your light bulbs? Your car?  Your heating system?
- Who do you trust to provide updates? Today? Tomorrow? In 10 years?
  - Long term, only community maintenance *might* possibly succeed
  - *Binary blobs leave you helpless and vulnerable, forever*

# Home Routers, Modems, etc.

- Most important, as they are both MITM and your lifeline
- Brand new devices unmaintained and unpatched
  - **New devices start with 4 year old code!**
- Firmware is usually not updated after ~1 year after sale by vendor, after the crash rate diminishes, then rots
- Embedded devices (e.g. your Nest thermostats) are no different than routers, except they are not on your path to the rest of the world...
- We now depend on our Internet service
  - e.g. POTS (wired telephones) are doomed: you'd like basic things like your phone to work in an emergency

# Wake Up Calls

- Research demonstrating *single* vulnerabilities that affect > half of the tested home routers
- DNSchanger attacked home routers as well as hosts
- 4.5 million DSL routers in Brazil
- TheMoon worm: most models of Linksys routers
- Heartbleed… Shellshock...
- It's a matter of when, rather than if, we have a big, big problem, if we don't already...

# Nightmare Scenarios

- *The Nightmare on Connected Home Street* is amusing but is actually only a minor bad dream
- Here's a *real* nightmare as likely as Honan's...
  - The broadband edge of the Internet stops working one day, and cannot be resurrected even by a power cycle
  - You have no Internet access at all, and cannot access patches for **anything**
  - Devices might even need to be replaced
  - I've lived this nightmare. I looked over into the abyss... Root cause: binary blob in **other** non-upgradeable devices. Whether implant, malware, or bug, it is still a disaster...

# Hysterical Causes

- First Linux flash file system elided UID's/GID's
  - Everything typically runs as “root”
  - Many minor vulnerabilities therefore become major vulnerabilities
  - “Simple” matter of configuration to fix this problem, but someone has to do the work!
- Flash size has prevented use of “upstream” distro's that have ongoing security updates and upgrades
  - Would love to have current volume bill of materials data: at some date, the tiny flash devices cost more than modern large ones
- Economic disconnect between “costs” and “benefits”
- BOM costs drives boot loader to unprotected flash
- Binary blob disaster

# Binary Blob Disaster

- Silicon vendors design a board support package for their silicon, to be shopped to ODM's to shop for possible “design wins”
  - Silicon vendor freezes on a static version of Linux/applications
  - Write a device driver, usually a “binary blob”, despite the GPL
  - Or the chip/module has a processor, with its own OS/binary blob
  - Even if the vendor has not been suborned by an intelligence agency, the code for these blobs we've usually seen is poor
- ODM's often ship that code, “with sugar on top”
- No incentive for updates, until the next generation silicon is being shopped around again: years later, maybe they do something
- Net result: No update stream and a frozen ecosystem: even if ODM's had the expertise and incentive, ODM's **cannot** update to current software to fix vulnerabilities, and 3rd parties cannot maintain it either
  - And ODM's have no financial incentive to update, either

# Disconnect of Incentives: Supplier

- Silicon vendor: incentive for design wins, software is an expense and an after-thought
  - Vulnerabilities occur much later
- ODM: if it doesn't crash, and doesn't get bad reviews while the device is for sale, they are happy:
  - Vulnerabilities come later
  - They have little software expertise, and what they have goes to what they consider market differentiation, since the silicon vendors prevent serious updates: race to the bottom
- Efforts are not pooled between ODM's since the code base is too old to integrate into upstream projects
- Updates are left to user installation, if ever published

# Disconnect of Incentives Consumers

Depending on market, devices are purchased by either ISP's or end-users, who “pay” the support costs

- Installation costs usually exceeds the purchase costs

- ISP's
  - Home routers sometimes provided by your ISP
  - No competence in security in devices: e.g. British Telecom's “backdoor”. ISP's goals differ from their customers
- End users
  - It is difficult/impossible for you to buy a “better” device
  - They (you) don't know enough to differentiate the devices beyond “crash” bugs, so you buy on price alone (Consumer Reports for home routers???): race to the bottom

# Pernicious Effects of Blobs on the Ecosystem

- Device vendors cannot work effectively in the Linux ecosystem
  - Therefore embedded systems vendors do not
  - Vendors support expenses are higher, in a market that is already a race to the bottom: they cannot share continuing engineering for their base systems (if they had the margins, which they don't)
  - Patches for bugs against antique code cannot go “upstream” to maintainers
  - Firmware cannot be easily updated to current code
  - Vendors typically can only abandon the firmware
- Alternate firmware is difficult/impossible, or doomed to also be stale
  - Alternate OS platforms difficult to impossible
  - Encouraging monoculture

# Devices need a secure boot loader

- Else you have no place to stand for security
- Users must be able to unlock them!
  - due to disaster, support & trust scenarios
- Cost is minimal: between $.00 and $.28 cents
  - Depending if the flash has proper locking facilities
  - If not, you need one D flop and a separate boot room
- OpenFirmware does all the required crypto, etc; no code need be developed

# Actions Needed (Technical)

- Fork lift upgrade of the entire edge of the Internet is the only solution!
- Secure the bootloader (properly); owners/users must be able to unlock them at will
- Apply existing technologies to build less insecure systems
- Disaster planning
- But it will all be for naught, if we do not re-engineer our business and software engineering practices

# Business Problem is Really Hard The Ecosystem Must Change...

- Fundamentally, a tragedy of the commons problem
  - How to connect those with the money (e.g. the users and ISP's), with those who develop and maintain the code?
  - Need funding model for continuing engineering of these devices for their lifetime
  - Need organizations to share software engineering among ODM's (Original Device Manufacturers)
  - Avoid monoculture....
- “Proprietary” information: e.g. binary blobs, documentation
- Who do you trust?
  - No single solution will suffice, not a vendor, ISP, or otherwise
- Collective actions are necessary

# Surprising Result

- RMS was right about Tivoization!
  - Says me, who helped define the MIT License
- But he had only two of the three real reasons:
  - *Life*
  - Liberty
  - The Pursuit of Happiness
- Whether this is enforceable by software license is orthogonal to the basic principle....

# Is There Hope?

- Some:
  - Linux Foundation
  - Embedian
  - OpenWrt
  - Other community efforts
- But closed binary driver and firmware blobs and lack of documentation limit and fragment the effectiveness of these efforts
- Some ISP's are aware that the market serves them very poorly...

# Open source router projects

- The most interesting (by far) is OpenWrt
  - Keeps up to date with upstream projects
  - Runs on >150 models of home routers (a fraction of the market though...)
  - Used at serious volume and is the “upstream” for many of the smaller commercial vendors (e.g. Fon, Buffalo, some of the Ubiquity devices, etc).
  - Large community is using OpenWrt in interesting ways at scale (the European community networking groups use OpenWrt as their common basis)
- Approximately 4 years ahead of the commercial markets, but badly needs resources applied...

# Some Policy Questions...

- How do we identify “critical infrastructure”? You can't predict it!
  - Anything that reaches sufficient volume can become critical infrastructure. Worse yet: you don't know in advance what will succeed in the market
  - The driver or firmware for a chip may be built into many different devices. Actual monocultures are often dismayingly widespread
- Code without a community is worthless.
  Must code be “born open”? How to handle organizational change over decades? How long must devices see support? Lifetime of devices? Should unmaintained devices “suicide”?
  - Code escrow is suspect at best...
  - How do you know you have the right code? Under what circumstances is it released? Will support/maintenance/build infrastructure be available when necessary to fix a critical problem? Will the people or organization even exist?
  - Devices can easily have lifetimes longer than most human organizations

# More Policy Questions...

- Mono-cultures are dangerous
  - Linux, many other software packages...
  - How do we encourage alternatives to Linux? Again, binary blobs & lack of documentation make achieving critical mass in alternatives (e.g. *BSD) to Linux very difficult
- “After market” upgrades
  - Should “Proprietary” information be able to inhibit other players moving into a market with upgrades, both hardware and software?

# More Policy Questions...

- How do we solve the tragedy of the commons problem? Who pays for development & support? How and why do people/organizations pay? You have to feed the penguins....
  - Critical infrastructure is any software widely enough used: e.g. OpenSSL/Heartbleed, but many more
  - Can be even a device driver or firmware for a device
- What role should industry and government have in pushing the market in a safer direction?

# A Meme to Spread

“Friends don't let friends run factory firmware”

Questions?

# What can you do today?

- Install OpenWrt/CeroWrt today and come help
  - Most of CeroWrt's developments are now upstream in OpenWrt
- If you are desiging devices
  - make availability of source for all components a priority in component selection: say no to binary blobs of any sort
  - Build in secure update facilities from the start
- If you are selecting devices, keep this all in mind

# CeroWrt: an advanced, bleeding edge build of OpenWrt

- Linux 3.10.44 kernel (currently)
- Platform for bufferbloat work: fq_codel, etc.
- Routes, rather than bridges the network devices
- Entropy!!! (at least > 0...)
- Mesh networking (Babel & others available in Quagga)
- IPv6 support, source sensitive routing (multiple upstream nets)
- Current dnsmasq w. DNSSec support
- Network test tools

# Come help!

- The network you secure may be your own
- OpenWrt has a user base of scale millions to convert to your ideas
- Your work may enter the mainstream market
- We have lots of security technology: but what should we actually apply here?