“The trouble with programmers is that you can never tell what a programmer is doing until it’s too late.”

– Seymour Cray
Security Mitigation & Validation

- Anti-Patterns for security mitigation & validation
  - Poorly considered password policy
  - Poorly considered privilege management
  - Assuming firewall or air gap is perfect security
  - No implementing secure update + secure boot
  - Just relying on penetration testing

- Mitigation best practices
  - Keep up to date with good security practices
  - Secure update + secure boot
  - Penetration testing is only a starting point
Principle: Password Strength

- **Typical failure scenarios**
  - Same password used by everyone
  - Weak passwords ("1234")
  - Strong password policy ➔ post-it note work-around

- **Possible solutions**
  - Different password per person with reasonable strength
  - Two-factor authentication (e.g., RFID transponder)

- **Balance between usability & security**
  - Can you memorize:  `7R#Ve9j3e@ahi7gjHr(*\pW4!X?`
  - 2017 NIST guidelines (https://pages.nist.gov/800-63-3/)
    - Good ideas: long size, hash/salt/stretch for storage
    - Avoid: words in dictionary, requiring weird characters, password hints, timed expiry
    - Avoid SMS for 2fa (!) due to phone number hijacking (at least in some countries)
Storing Passwords

- Don’t store them as plain text!
  - Don’t just encrypt them either

- Hash:
  - Store a digest of password
  - But, dictionary attacks are a problem
  - Rainbow table: precomputed hashes

- Salting & pepper:
  - Salt: random extra text
  - Pepper: systematic extra text
  - Can be secret or public (tradeoffs)

- Generically, key stretching:
  - E.g., PBKDF2 stretching
  - Use up to date techniques!
Rubber Hose Attack

A Crypto Nerd's Imagination:

His laptop's encrypted. Let's build a million-dollar cluster to crack it.

No good! It's 4096-bit RSA!

Blast! Our evil plan is foiled!

What Would Actually Happen:

His laptop's encrypted. Drug him and hit him with this $5 wrench until he tells us the password.

Got it.

Permanent link to this comic: https://xkcd.com/538/
Principle: Least Privilege

- Each user & task should only have as much capability as it needs
  - Commonly, “user,” “administrator,” “factory”
  - Better: per-user fine-gain bit map of function permission
  - Related: helpful to log who did what (forensics)

- Common mistakes
  - Make a common task high privilege
    - Everyone used to log in as admin for Windows
  - Give everyone the same password
    - Once someone has admin, can’t roll them back
  - Make risky operations too easy (no confirmation)

- In general, think through permissions
  - Customers may push back, but this is important
What Happens With Unsigned Updates

Infotainment-to-CAN Firewall

- CPU non-secured update
- Attackers reflashed firewall to access CAN

What Happens With Unsigned Updates

Hackers Remotely Kill a Jeep on the Highway—With Me in It

I was driving 70 mph on the edge of downtown St. Louis when the exploit began to take hold. Though I hadn’t touched the dashboard, the vents in the Jeep Cherokee started blasting cold air at the maximum setting, shifting the sweat on my back through the in-seat climate control system. Next the radio switched to the local hip hop station and began blaring “Kiss Me—At Full Volume.” I spun the control knob left and hit the power button, to no avail. Then the windshield wipers turned on, and wiper fluid coated the glass.

As I tried to cope with all this, a picture of the two hackers performing these stunts appeared on the car’s digital display. Charlie Miller and Chris Valasek, wearing their trademark ski suits. A nice touch, I thought.

http://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/
Secure Update

- You’ll need to deploy security patches
  - Your code might have a vulnerability
  - 3rd party code (library, OS, communications) might be vulnerable

- Secure update good practices:
  - Bootloader that does updates
    - First stage: integrity check for 2nd stage; can’t be changed(!)
    - Second stage: knows how to load application image
  - Bootloader checks image public key signature
    - Public key hard-coded into bootloader
    - Only properly signed images are loaded
    - Consider limited date ranges (key revocation is hard)
      » E.g., pre-deploy public key every 3 months for 20 years
    - Consider hard-coding repository IP addresses
If your firmware is compromised, you are insecure

- Need a way to make sure you only run factory-authorized code
- Use public key signature to check firmware image integrity
  - Note: symmetric hash exposes signing key to attack

Example Mitigation: Secure Boot

https://www.faa.gov/aircraft/air_cert/design_approvals/air_software/media/AR-08-31.pdf
Misconception: “Encryption Equals Security”

- Encryption provides secrecy – but you might need integrity!
- Encryption invokes export controls
- **What are the actual security requirements?**

Example for firmware distribution

- Public key encryption of firmware is infeasible
  - Need a different binary image for every device!
  - On-line copy vulnerable to attack
  - Reverse engineering will recover firmware image if bad guys want it
- **Secure signature (Public Key Digest) works well**
  - A digest is a small hash of the entire message (like a checksum, but crypto-secure)
  - Sign image off-line one time; all devices can use public key to validate
  - Use per-download encryption as defense in depth
“Pen test” – attempt to attack system to look for problems

- **Automated vulnerability testing**
  - Test known security exploits to see if they succeed
  - Test for bug fixes for known non-exploit vulnerabilities
  - Port scanning for dangerous open (unnecessary) Ethernet ports

- **Penetration analysis**
  - Hire a “red team” to attempt to penetrate system
  - Fuzz testing – send random inputs to see what breaks
Code Analysis

Static & dynamic code analysis

- General code quality tools: Coverity, PC-Lint
- Security-specific security tools
  - Look for violations of checkable secure coding rules
  - Various tools for thread safety, bounds checking, ...
- Potential problem:
  - False positives (many warnings are not actual vulnerabilities)

Peer review

- Security-oriented review of source code
- E.g., Cert C 98 Coding Standard
  - E.g., use `strcpy_s()` instead of `strcpy()`
Many Other Approaches

- **Intrusion detection**
  - Detect abnormal patterns of system operation
  - False positives are expensive; no such system is perfect

- **Monitor Black Hat sites**
  - Look for published exploits against your product

- **Honey pot systems**
  - Deploy a monitored decoy system and look for successful attacks

- **Bug bounties**
  - Pay anyone who finds an exploit so you can fix it
Security Mitigation & Validation

- **Good practices:**
  - Encourage strong but usable passwords
  - Use fine-grain permissions
  - Be careful storing password information
  - Respect limitations of firewall approaches
  - Use secure update and secure boot
  - Use more than just penetration testing

- **Pitfalls:**
  - Thinking security is easy
  - Using intuition instead of doing your homework
HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.

OH, DEAR — DID HE BREAK SOMETHING?

IN A WAY—

DID YOU REALLY NAME YOUR SON 'ROBERT'); DROP TABLE Students;-- ?

OH, YES. LITTLE BOBBY TABLES, WE CALL HIM.

WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.

AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

https://xkcd.com/327/