Nick's Personal Self-Defense Decisions...
Putting CS161 in Context: Nick's Self Defense Strategies...

• **How** and **why** do I protect myself online and in person...
  
• **How** I decide what to prepare for (and what not to prepare for)

• **Why** I've drunk the Apple Kool-Aid™

• **Why** I use my credit card everywhere but not a debit card

• And my future nightmares:
  
• What do I see as the security problems of tomorrow...
My Personal Threats: The Generic Opportunist

- There are a *lot* of crooks out there
  - And they are rather organized...
- But at the same time, these criminals are generally economically rational
  - So *this* is a bear race: I don't need perfect security, I just need *good enough* security
- I use this to determine security/convenience tradeoffs all the time
  - So no password reuse (use a password manager instead)
  - Full disk encryption & passwords on devices: Mitigates the damage from theft
  - Find my iPhone turned on: Increases probability of theft recovery
My Personal Threats: The *Lazy* Nation State

- OK, I'm a high **enough** profile to have to worry about the "Advanced Persistent Threats"...
- Trying for a reasonably high profile on computer policy issues
- A fair amount of stuff studying the NSA's toys and other nation-state tools
- But only at the Annoying Pestilent Teenager level: I'm worth some effort but not an extraordinary amount
- So its only **slightly** more advanced than the everyday attackers...

**With one huge exception: Crossing borders**

- Every nation maintains the right to conduct searches of all electronic contents at a border checkpoint
My Border Crossing Policy: Low Risk Borders

- Not very sensitive borders: Canada, Europe, US, etc...
  - I use full disk encryption with strong passwords on all devices
    - Primary use is to prevent theft from also losing data
  - I have a **very robust** backup strategy
    - Time machine, archived backups in a safe deposit box, working sets under version control backed up to remote systems...

- So, as the plane lands:
  - Power off my devices
    - Device encryption is only **robust** when you aren't logged in
  - Go through the border

- If my devices get seized...
  - "Keep it, we'll let the lawyers sort it out"
High Risk Borders

• Middle East or, if, god forbid, I visit China or Russia...
  • Need something that doesn't just resist compromise but can also *tolerate* compromise

• A "burner" iPhone SE with a Bluetooth keyboard
  • The cheapest secure device available
  • Set it up with *independent* computer accounts for both Google and Apple
    • Temporarily forward my main email to a temporary gmail account
    • All workflow accessible through Google apps on that device
  • Bluetooth keyboard does leak keystrokes, so don't use it for passwords but its safe for everything else

• Not only is this device very hard to compromise...
  • But there is very low value in *successfully compromising it*: The attacker would only gain access to dummy accounts that have no additional privileges

• And bonus, I'm not stuck dragging a computer to the ski slopes in Dubai...
  • Since the other unique threat in those environments is the "Evil maid" attack
My Personal Threats:
The Russians...  Perhaps

• This is the paper that killed the Viagra® Spam business
• A $100M a year set of organized criminal enterprises in Russia...
  And they put the organized in organized crime...

• I've adopted a detection and response strategy:
  • The Russians have higher priority targets: The first authors, the last authors, and Brian Krebs
  • If anything suspicious happens to Brian, Kirill, or Stefan, then I will start sleeping with a rifle under my bed
The Apple Kool-Aid...

- The iPhone is perhaps the most secure commodity device available...
  - Not only does it receive patches but since the 5S it gained a dedicated cryptographic coprocessor

- The **Secure Enclave Processor** is the trusted base for the phone
  - Even the main operating system isn't fully trusted by the phone!

- A dedicated ARM v7 coprocessor
  - Small amount of memory, a true RNG, cryptographic engine, etc...
  - Important: A collection of *randomly* set fuses
    - Should not be able to extract these bits without taking the CPU apart or compromising the Secure Enclave's software
  - But bulk of the memory is shared with the main CPU
The Roll of the SEP...

Things **too important** to allow the OS to handle

- Key management for the encrypted data store
  - The CPU has to ask for access to data!
- Managing the user's passphrase and related information
- User authentication:
  - *Encrypted* channel to the fingerprint reader/face recognition camera
- Storing credit cards
  - ApplePay is cheap for merchants *because it is secure*:
    Designed to have very low probability of fraud!
AES-256-XEX mode

- An *confidentiality-only* mode developed by Phil Rogaway...
  - Designed for encrypting data within a filesystem block \( i \)
  - Known plaintext, when encrypted, can't be replaced to produce known output, only "random" output
  - \( \alpha \) is a galios multiplication and is very quick:
    - In practice this enables parallel encryption/decryption

- Used by the SEP to encrypt its own memory...
  - Since it has to share main memory with the main processor

- Opens a limited attack surface from the main processor:
  - Main processor can replace 128b blocks with random corruption
User Passwords...

- Data is encrypted with the user's password
  - When you power on the phone, most data is completely encrypted
- The master key is PBKDF2(password || on-chip-secret)
  - So you need both to generate the master key
  - Some other data has the key as F(on-chip-secret) for stuff that is always available from boot
- The master keys encrypt a block in the flash that holds all the other keys
  - So if the system can erase this block effectively it can erase the phone by erasing just one block of information
- Apple implemented effaceable storage:
  - After x failures, OS command, whatever...
    Overwrite that master block in the flash securely
  - Destroy the keys == erase everything!
Background: FBI v Apple

- A "terrorist" went on a rampage with a rifle in San Bernardino...
  - Killed several people before being killed in a battle with police
- He left behind a work-owned, passcode-locked iPhone 5 in his other car...
- The FBI *knew* there was no valuable information on this phone
  - But never one to refuse a good test case, they tried to compel Apple in court to force Apple to unlock the phone...
- Apple has serious security on the phone
  - Effectively everything is encrypted with PBKDF2(PW||on-chip-secret):
    - >128b of randomly set microscopic fuses
    - Requires that *any* brute force attack either be done on the phone or take apart the CPU
  - Multiple timeouts:
    - 5 incorrect passwords -> starts to slow down
    - 10 incorrect passwords -> optional (opt-in) erase-the-phone
What the FBI wanted...

• Apple provides a **modified** version of the operating system which...
  • Removes the timeout on all password attempts
  • Enables password attempts through the USB connection

• **Apple cryptographically signs the rogue OS version!**
  • A horrific precedent: This is **requiring** that Apple both create a malicious version of the OS and sign it
    • If the FBI could compel Apple to do this, the NSA could too... It would make it **impossible** to trust software updates!
Updating the SEP To Prevent This Possibility...

- The SEP will only accept updates *signed by Apple*
  - But an updated SEP could exfiltrate the secret to enable an offline attack
- The FBI previously asked for this capability against a non-SEP equipped phone
  - "Hey Apple, cryptographically sign a corrupted version of the OS so that we can brute-force a password"
- How to prevent the FBI from asking again?
  - Now, an OS update (either to the base OS and/or the SEP) requires the user to be logged in *and input the password*
    - "To rekey the lock, you must first unlock the lock"
    - The FBI can only even *attempt* to ask before they have possession of the phone since once they have the phone they must also have the passcode
    - So when offered the chance to try again with a "Lone Wolf's" iPhone in the Texas church shooting, they haven't bothered
The Limits of the SEP...

The host O/S

- The SEP can keep the host OS from accessing things it shouldn't...
  - Credit cards stored for ApplePay, your fingerprint, etc...
- But it can't keep the host OS from things it is supposed to access
  - All the user data when the user is logged in...
- So do have to rely on the host OS as part of my TCB
  - Fortunately it is updated continuously when vulnerabilities are found
    - Apple has responded to the discovery of very targeted zero-days in <30 days
  - And Apple has both good sandboxing of user applications and a history of decent vetting
    - So the random apps are not in the Trusted Base.
The SEP and Apple Pay

- The SEP is what makes Apple Pay possible
  - It handles the authentication to the user with the fingerprint reader/face reader
    - Verifies that it is the user not somebody random
  - It handles the emulation of the credit card
    - A "tokenized" Near Field Communication (NFC) wireless protocol
    - And a tokenized public key protocol for payments through the app

- **Very hard** to conduct a fraudulent transaction
  - Designed to enforce user consent at the SEP

- **Disadvantage**: The fingerprint reader is part of the trust domain
  - Which means you need special permission from Apple to replace the fingerprint reader when replacing a broken screen
I love ApplePay...

- It is a **faster** protocol than the chip-and-signature
  - NFC protocol is designed to do the same operation in less time because the protocol is newer
- It is a **more secure** protocol than NFC on the credit card
  - Since it actually enforces user-consent
- It is more **privacy sensitive** than standard credit card payments
  - Generates a unique token for each transaction:
    Merchant is not supposed to link your transactions
- Result is its low cost:
  - Very hard to commit fraud -> less cost to transact
- I use it on my watch all the time
- Useful product idea: Enable enrolling credit cards to enable "tap to open" door locks!
Transitive Trust in the Apple Ecosystem...

- The most trusted item is the iPhone SEP
  - Assumed to be rock-solid
  - Fingerprint reader allows it to be convenient
- The watch trusts the phone
  - The pairing process includes a cryptographic key exchange mediated by close proximity and the camera
  - So Unlock the phone -> Unlock the watch
- My computer trusts my watch
  - Distance-bounded cryptographic protocol
  - So my watch unlocks my computer
- Result? I don't have to keep retyping my password
  - Allows the use of strong passwords everywhere without driving myself crazy!
Credit Card Fraud

• Under US law we have very good protections against fraud
  • Theoretical $50 limit if we catch it quickly
  • $0 limit in practice
• So cost of credit card fraud for me is the cost of recovery from fraud
  • Because fraud *will happen*:
    • The mag stripe is all that is needed to duplicate a swipe-card
      • And you can still use swipe-only at gas pumps and other such locations
    • The numbers front and back is all that is needed for card-not-present fraud
      • And how many systems
• What are the recovery costs?
  • Being without the card for a couple of days...
    • Have a second back-up card
  • Having to change all my autopay items...
    • Grrrr....
But What About "Debit" Cards?

• Theoretically the fraud protection is the same...

• But two caveats...

  • It is easier to not pay your credit card company than to claw money back from your bank...

  • Until the situation is resolved:
    • Credit card? It is the credit card company's money that is missing
    • Debit card? It is your money that is missing

• Result is debit card fraud is more transient disruptions...
So Two Different Policies...

- **Credit card: Hakunna Matata!**
  - I use it without reservation, just with a spare in case something happens
  - Probably 2-3 compromise events have happened, and it's annoying but ah well
  - The most interesting was $1 to Tsunami relief in 2004... was a way for the attacker to test that the stolen card was valid

- **Debit card: Paranoia-city...**
  - It is an ATM-ONLY card (no Visa/Mastercard logo!)
  - It is used ONLY in ATMs belonging to my bank
    - Reduce the risk of "skimmers": rogue ATMs
Nick's Nightmare: Slaughterbots™

- Take a toy drone chassis design
  - <$40 retail price!
- Add two cameras...
  - Enables stereo vision for navigation & targeting
- Add a Zynq FPGA and a single RAM chip
  - Gives a dual-core ARM CPU, a significant amount of FPGA resources, and 1 GB RAM
- Add a miniature EFP (Explosively Formed Penetrator/Explosively Formed Projectile)
  - Explodes and turns a metal disk into effectively a bullet without the need for a barrel
  - Or could just do an electronically-fired derringer design with an integrated bullet/barrel
Back of the Envelope Design Costs...

- $10M R&D budget
  - Develops mini-EFP, circuit board, and autonomous software
- $200/each production cost
  - Cost over toy drone:
    EFP, control board w FPGA & memory, swap Lithium Ion (rechargeable) battery with standard Lithium battery (more energy density)
- Also $500-1000 "carrier drones"
  - Fixed-wing mother-drone for longer-range delivery:
    single larger motor, two servos, same computer with the addition of a GPS
    - Fly to specified location, drop the Slaughterbots...
So the HARD challenge: How to *stop* these things in a city!

- Can't just blast away with bullets or lasers...
  - After all, what happens when you miss?
- Can't use some super sekret military technology
  - You can't put classified stuff all over the place
- Can't use something super expensive...
  - We need to cover a lot of territory cheaply
- So it is an interesting hard problem to think about...