Module 0.4
What is Computer Security and What to learn?

Ken Thompson’s clever Trojan

- Ken Thompson, co-author of UNIX, recounted a story of how he created a version of the C compiler that, when presented with the source code for the "login" program, would automatically compile in a backdoor to allow him entry to the system. This is only half the story, though. In order to hide this Trojan horse, Ken also added to this version of "cc" the ability to recognize if it was recompiling itself to make sure that the newly compiled C compiler contained both the "login" backdoor, and the code to insert both Trojans into a newly compiled C compiler. In this way, the source code for the C compiler would never show that these Trojans existed.

What is Security?

- Achieving something in the presence of adversaries
  - Internet is full of adversaries
  - There are insider adversaries for air-gapped systems
  - Thus design of systems need to worry about security

- A High Level Plan for Security Centric System Design
  - Policy: “Only X can access file F”
  - Common goals: Confidentiality, Integrity, Availability
  - Threat Models: “Can Y physically grab the file server?”
  - Mechanisms: The knobs that can be controlled to uphold your security policy, but also be flexible to uphold a different policy
  - Resulting Goal: “No way the adversary in the threat model to violate policy”

Why is security hard?

- Need to guarantee policy, assuming threat models
- Difficult to think of all possible ways that attacker might break in
- Realistic Threat models are open-ended (Negative models)
- Easy to check a positive goal (“X has access to File F”)
- Weakest link matters
- Iterative process: Design, Update Threat Model as necessary, assess vulnerability → Design, Update …..

What if perfect Security is not achievable?

- Best effort
- Each system will have some breaking point – need to analyze and understand – e.g., penetration testing
- Need to manage security risk vs. benefit tradeoff
- Risk based security model
- Manual auditing often can help
- Make the cost of attack high – deterrence
  - Either by law
  - Technologically

Why Policy matters in Security

- Example: Sarah Palin's email account hacked
  - Yahoo accounts have username/password and security questions
  - User can login with username/password
  - If user forgets password – can reset by answering security question
  - Security questions are sometimes easier to guess
  - Some one guessed Palin's highschool, birthday etc
  - Policy amounts to: can log in with either password or security questions

Policy Matters: Example 2

- Mat Honan's accounts at Amazon, Apple, Google etc hacked
  - Gmail password reset: send a verification link to a backup email
    - Google helpfully prints part of the backup email address
    - Mat Honan's backup address was his Apple @me.com account
  - Apple password reset: need billing address, last 4 digits of credit card
    - Address can be easily found, how do you get last 4 digits of credit card
  - Amazon: can add a credit card to an account, no password required
    - Amazon password reset: provide any of user's credit card number
    - Amazon will not print credit card number but print last 4 digits

What to do?

- Think hard about implications of policy statements
- Some policy checking tools can help – but you need to specify ‘what is bad’
- Difficult in distributed systems: don’t know what everyone is doing

What might go wrong in threat models/assumptions?

- Human factors not accounted for: ex. Phishing attack
- Computational assumptions change over time:
  - MIT's kerberos system used 56-bit DES keys since mid 1980s
  - Now it costs about $100 to get it cracked
- All SSL certificate CAs are fully trusted
  - To connect to an SSL-enabled website, your browser verifies the certificate
    - Certificate is a combination of server's host name, and cryptographic key, signed by a trusted CA
  - 100s of CAs are trusted by most browsers
  - In 2011, two CAs were compromised – issued fake certificates for many domains (google, yahoo, tor, …)
  - http://en.wikipedia.org/wiki/Comodo_Group
  - http://en.wikipedia.org/wiki/DigiNotar

Limitations in Assumptions

- Assuming your hardware is trustworthy
  - If NSA is your adversary – it is not necessarily true
    - https://www.schneier.com/blog/archives/2013/12/more_about_the.html
- Assuming good randomness in cryptography
  - Often source of randomness may not be good, and keys may be compromised
  - https://factorable.net/weakkeys12.extended.pdf
- Assuming OS to be secure
  - Backdoors? Trojans?
- Machine is disconnected from the Network
  - Did not stop stuxnet worm

What to do to avoid limitations in threat models?

- More explicit and formalized threat models to understand possible weaknesses
- Simpler and more general threat models
- Better design may lessen reliance on certain assumptions
  - E.g., alternative trust models that does not rely on full trust in CAs
  - E.g., authentication mechanisms that aren't susceptible to phishing

Problems with mechanisms

- Bugs in security mechanism (e.g. OS kernel) lead to vulnerabilities
- If application is enforcing security, application bugs can lead to vulnerabilities
- Example: Apple's iCloud password guessing rate limits http://thenextweb.com/apple/2014/09/01/this-could-be-the-apple-icloud-flaw-that-led-to-celebrity-photos-being-leaked/
- Example: Missing access control checks in Citigroup's credit card website http://www.nytimes.com/2011/06/14/technology/14security.html?_r=0
- Example: Android's Java SecureRandom weakness leads to bitcoin theft – the randomization seed was not being changed sometimes leading to easy guess of private keys

Some implementation bugs

- Buffer overflow
- Use-after-free (e.g., dereference a already deallocated pointer)
- Double-free
- Decrementing stack pointer past the end of stack – into some other memory location
  - http://www.invisiblethingslab.com/resources/misc-2010/xorg-large-memory-attacks.pdf
- Not checking sanity of inputs
  - Sql injection
  - Command injection