Introduction to Systems Security

51.502 Systems Security
Paweł Szałachowski

About Me

- ISTD's faculty member since Aug 2017
- Research and systems building
  - Blockchain and Internet-level Consensus
  - Public-key Infrastructures, SSL/TLS, Internet security…
  - SCION Internet Architecture: https://www.scion-architecture.net/
- https://pszal.github.io ; pawel@sutd.edu.sg ; 1.402-35

Organization

- 12 credits
  - 3/3/6
- TA/tutor
  - Dr. Ivan Homoliak (ivan_homoliak@sutd.edu.sg)
- Ubuntu 16.04 x86_64, Python, C, PHP, JS, …
- Grading
  - 50% homework (**NO** deadline extensions, 2 students per group)
  - 50% final

Reading List

Concrete chapters to read will be given at the end of every lecture.

- [And] Ross Anderson, Security Engineering, John Wiley & Sons, 2001

- [BSS] David Basin, Michael Schläpfer, and Patrick Schaller, Applied Information Security: A Hands-on Approach, Springer, 2011

- [FSK] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno, Cryptography Engineering, Wiley, 2012.

- [GT] Michael T. Goodrich and Roberto Tamassia, Introduction to Computer Security, 2011

- [Sch] Schneier, Bruce. Applied Cryptography: Protocols, Algorithms, and Source Code in C. John Wiley & Sons, 1996

- [Sho] Adam Shostack "Threat modeling: designing for security", Wiley, 2014

- [Sta] Mark Stamp, Information Security: Principles and Practice, John Wiley & and Sons, 2006

Systems Security

- Why?!
- Systems are everywhere
- Systems
  - Full stacks, multiple layers, languages, implementations, OSes, HW, … interacting together
  - Difficult to manage, maintain, integrate, …
  - Legacy systems

Systems Security

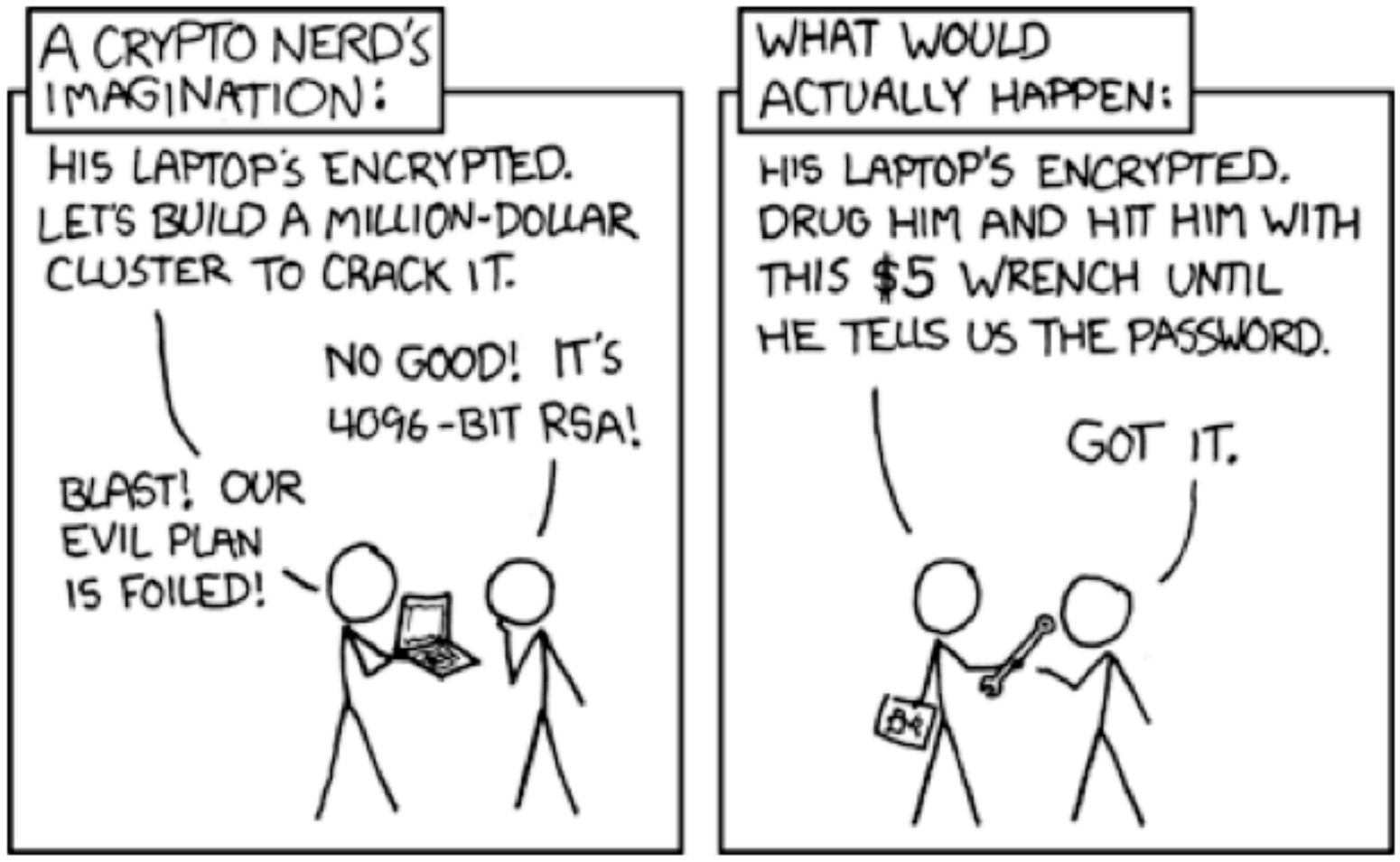
- Secure system
    - Achieves goals in the presence of an adversary
- A goal is to successfully implement a security policy
    - e.g., provide confidentiality, integrity, availability, …
- Threat model
    - Assumptions about the adversary
- Mechanisms
    - Tools that the system provides to help implement (and maintain) the policy
- Success
    - The adversary (within the threat model) cannot violate the policy

Systems Security

- Why it is difficult to build secure systems?
    - Security is a negative goal
        - Protect all possible ways vs. find one way to attack
    - Threats models in reality are hard
        - Difficult to formalize
    - Security is a process
        - Not a one-time action, rather an iterative process
    - Business/User requirements

Threat Modeling

- System mc
  - Vulnera
  - Threat:
  - Attack:
  - Exploit:
  - Trust b
- Threat moc
  - Determ
  - Assign risk to the various threats
  - Drive the vulnerability mitigation process



Security Principles

- **Simplicity**
  - KISS: “Keep it Simple, Stupid”, “Keep it Simple & Secure”
  - Applies to any design, engineering, and implementation task
  - Simpler solutions are easier to understand, analyze, implement, maintain, …
  - Complexity is always an enemy of security

Security Principles

- **Open Design**
  - Security of a system should not depend on the secrecy of its protection mechanisms
  - Similar to Kerckhoffs' principle in cryptography
  - Secrets are difficult to protect
    - Thus amount of secret information should be reduced to a minimum

Security Principles

- **Compartmentalization**
  - Organize resources into isolated groups of similar needs.
    - Sometimes groups are called compartments, or zones, or domains, …
  - Used in many fields (services, networks, processes, OS architecture, languages)
  - Groups are easier to manage (access control, resources)
  - Attacks and failures are scoped to groups
  - Groups can have different policies and security measures
  - Difficult to implement (groups interact and need secure interfaces)

Security Principles

- **Minimum Exposure**
  - Minimize the attack surface a system presents to the adversary
  - Limit the amount of information given away
  - Reduce external interfaces to the necessary minimum
  - Minimize the window of opportunity for na adversary
    - Time available for an attack
    - Frequency of critical events

Security Principles

- **Least Privilege**
  - Any component and user of a system should operate using the least set of privileges necessary to complete its job
  - Reduce privileges to the absolute minimum
    - Actors of a system should not be allowed to access any objects and resources other than really required to complete their jobs
  - Helps to minimize negative consequences of attacks and operation errors
  - Difficult to implement
    - Security policies are not clearly defined and fine-grained

Security Principles

- **Minimize Trust and Maximize Trustworthiness**
  - Trusted vs. trustworthy system
    - User expectations, about system's behavior, are assumed to be satisfied (trusted) or are satisfied (trustworthy)
  - Minimizing trust is about minimizing assumptions
  - Maximizing trustworthiness is about turning assumptions into validated properties
  - Trust is a transitive relation

Security Principles

- **Secure and Fail-Safe Defaults**
  - System should start in and return to a secure state in the event of a failure
  - For instance, a security mechanism can be enabled at system start-up and re-enabled whenever the system (or its subsystem) fails
    - Whitelist vs. blacklist access control

Security Principles

- **Complete Mediation**
  - Access to any object must be monitored and controlled
  - Access control must be operational during lifetime
    - Operation, maintenance mode, shutdown, failure
  - Protection at a correct layer (layer-below attacks)

Security Principles

- **No Single Point of Failure**
  - Build redundant security mechanisms whenever feasible
  - If one mechanisms fails, there are others in place to prevent attacks
  - Sometimes (almost always;-) requirements limit security mechanisms
    - Usability, financial resources, performance, ….
  - SPOFs in security vs. SPOFs in availability

Security Principles

- **Traceability**
  - Log security-relevant system events
    - Detect errors and attacks
    - Relevant and complete information
    - Keep logs confidential, authentic, available
    - Often required by law
      - Some data should be stored anonymously

Other Security Principles

- **Generating Secrets**
  - Maximize the entropy of secrets
- **Usability**
  - Design usable security mechanisms and interfaces
- **...**

Examples


VS
Blend

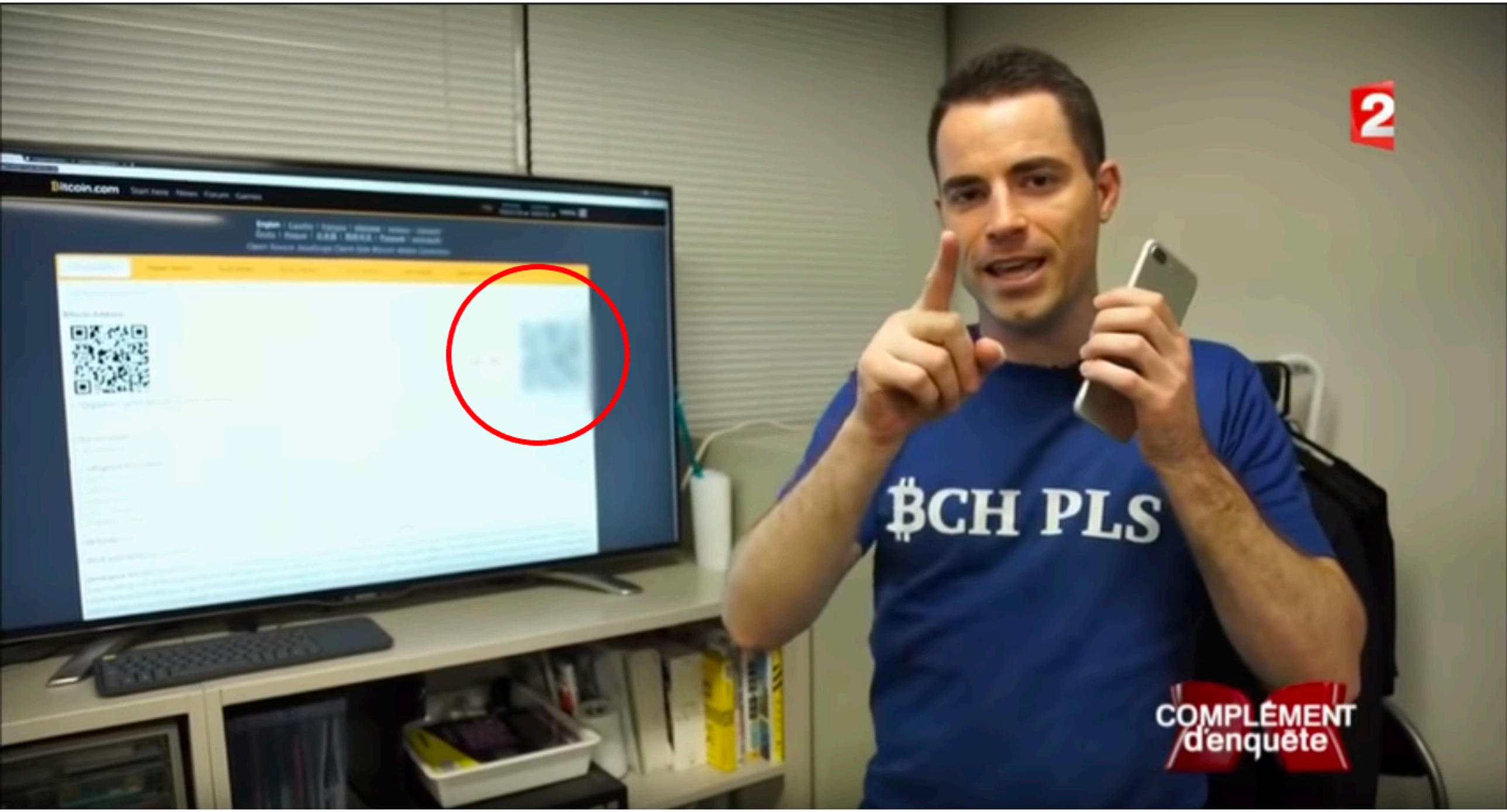

MAXIMUM SECURITY ENTRANCE


HID
2
3
6
7
8
*
0
#


PELCO


SECURITY WILL
RETURN IN
5 MINUTES
FAIL
failblog.org


Y U NO go to FAILKING.COM


2
₿CH PLS
COMPLÉMENT
d'enquête

Examples

- Password recovery questions
    - Account recovery mechanisms
- Time of Check to Time of Use (TOCTTOU)
- Buffer overflows
- SQL Injections
- …

OS-level Access, Privileges, Isolation, Sandboxing, …

Privilege Separation

- How to limit impact of a potential bug?
  - Divide system into modules
- **Unix/Linux**
  - Users, Processes, Files, I/O, CPU & mem, …
- VMs and containers
- Networks
- …

Unix: privilege model

- Principal (user)
  - Identified by user id (uid) and group id (gid)
    - /etc/passwd, /etc/group, /etc/shadow
    - Defines what privileges does process have
- Subject
  - Process (uid and gid(s))
- Objects that process may act on
  - Files/Directories, Memory, Process, I/O, IPC, …

Users

- Need for separating users
  - Own directory
    - Own uid (and usually own gid)
- Root user
  - Superuser with uid=0

Files & Directories

- What kind of privileges?
  - Files
    - read, write, exec, change permission, …
    - uid and gid associated
  - Directories
    - remove (unlink), link, rename, create, …
- rwx (for owner, group, and other)
  - -rw-rw-r-- 1 pawel pawel 1161 Jan 9 17:40 client.py
- Who can change permissions?
  - if file’s uid (owner) == user’s uid OR uid=0
- File descriptors (more granular access)

Processes

- Create
  - New process will get the same uid
- Kill
  - To kill process user has to have the same uid OR uid=0
- Debug
  - … the same uid OR uid=0
- Changing uid
  - setuid(new_id)
  - Only root (uid=0) can do this

Networking

- Listening
    - Only root (uid=0) can listen on ports < 1024
- Connecting
    - Anyone can connect
- Read/Write
    - Only to own sockets
- Send raw packets
    - Root required

Misc

- setuid binaries
  - /bin/passwd
  - Security issues
    - Bugs
      - LD_PRELOAD
- chroot
  - Root required
- Capabilities
  - Fine-grained control over root permissions
- SELinux
  - Mandatory Access Control (MAC) for Linux

Reading

- [And] chapters 4.1, 4.2, and 8.5
- [BSS] chapters 1 and 4
- [Sho] chapters 1 and 2