Threat modeling and Security architectures

Debdeep Mukhopadhyay
Mainack Mondal

CS 60065
Autumn 2020

Roadmap

- Basic security analysis
  - Threat modelling
  - Adversary modelling
- Design principles for security

How to do basic security analysis for a system?

- Question: Is a given system secure OR how do you secure the system?
  - *What* do we intend to protect? (system model)
  - *Who* is the attacker or the threat? (threat model)
  - What are the security requirements? (Security Goals)
  - What security approaches can be effective? (Solution)

1. System model

- Understand architecture of the system
- Enumerate asset and their value in the system
- Possible questions you should ask:
  - What are the exact assets? (be as specific as possible)
  - What is the operating value (can be $, can be man hours)
  - What is the impact if this asset if breached?

Question: What is the system model for protecting against password guessing attack on a banking website?

2. Threat modelling

- Identify potential attackers (script-kiddies, hacker-for-hire, your ex, a nation state?)
- Enumerate attacker resources
- Estimate number of attacks, probability of attack

Common adversary types / adversary attributes in threat models

- Attacker action
  - Passive (eavesdropping), Active (man-in-the-middle attack)
- Attacker capability
  - Script kiddies to nation states (decides how resilient your solution should be)
- Attacker access
  - External (can only observe the system), Internal (inside the system, e.g., compromised user account)

Common adversary types / adversary attributes in threat models

- Attacker action
  - Passive (eavesdropping), Active (man-in-the-middle attack)
- Attacker capability
  - Script kiddies to nation states (decides how resilient your solution should be)
- Attacker access
  - External (can only observe the system), Internal (inside the system, e.g., compromised user account)

Question: What is the threat model for protecting against password guessing attack on a banking website?

Schema for modelling adversaries

- Named group of adversaries (categorical schema)

	Named group of adversaries
1	Foreign intelligence (including government-funded agencies)
2	cyber-terrorists or politically-motivated adversaries
3	industrial espionage agents (perhaps funded by competitors)
4	organized crime syndicate
5	lesser criminals and crackers (i.e., individuals who break into computers)
6	malicious insiders (including disgruntled employees)
7	non-malicious employees (often security-unaware)

Schema for modelling adversaries

- Named group of adversaries (categorical schema)

	Named group of adversaries
1	Foreign intelligence (including government-funded agencies)
2	cyber-terrorists or politically-motivated adversaries
3	industrial espionage agents (perhaps funded by competitors)
4	organized crime syndicate
5	lesser criminals and crackers (i.e., individuals who break into computers)
6	malicious insiders (including disgruntled employees)
7	non-malicious employees (often security-unaware)

Also targeted vs. generic attacks

Threat modelling

Approaches

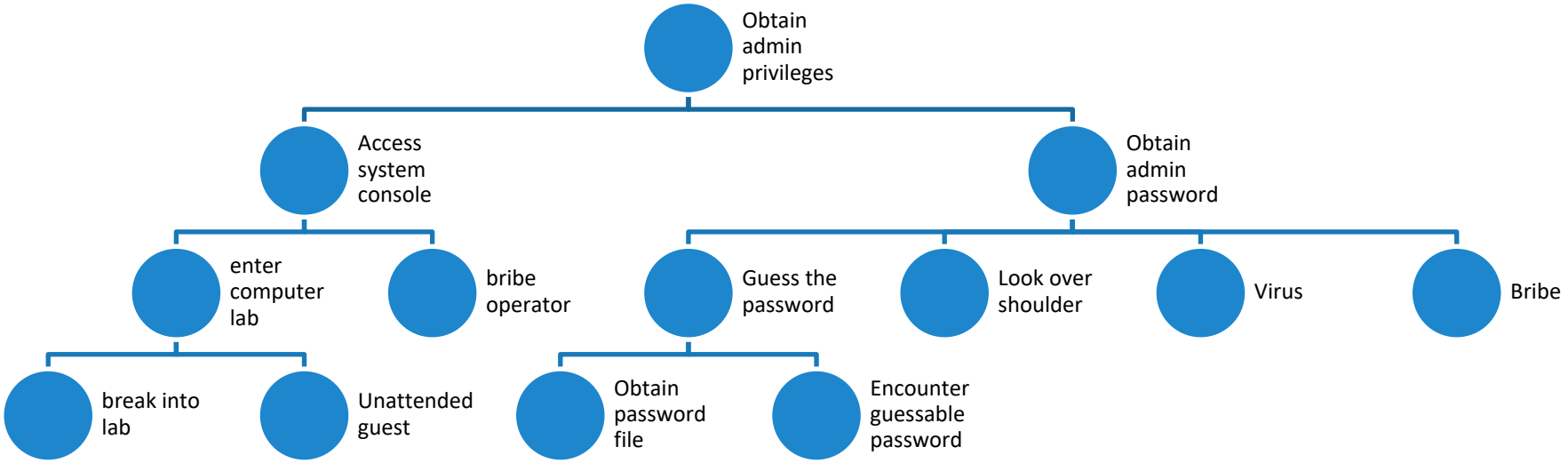
- Architectural diagrams (data-flow, user-flow diagram)
- Attack trees (creating possible attack vectors)
- Checklists
- STRIDE

Threat modelling

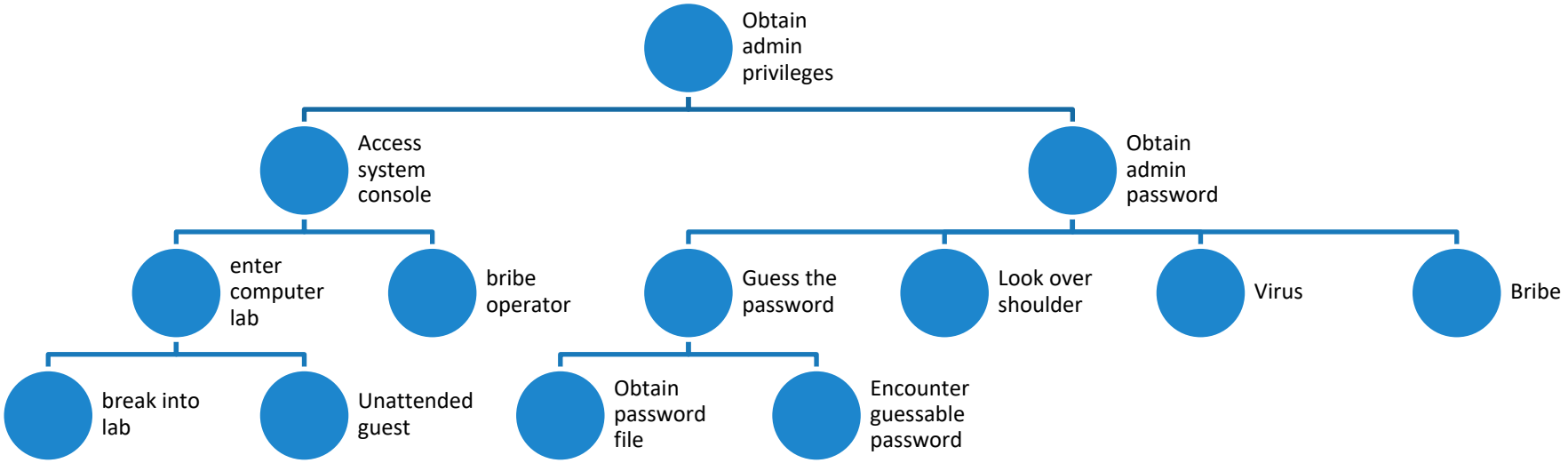
Approaches

- ~~Architectural diagrams (data-flow, user-flow diagram)~~
- Attack trees (creating possible attack vectors)
- Checklists
- STRIDE

Example: Attack tree



Example: Attack tree



Practical example: https://tools.ietf.org/html/draft-convery-bgpattack-01#page-6

Checklists

Laundry list of all possible attacks

- Good basic choice
- The security tester need to adapt to the tested system

STRIDE

Look for the following attacks:

- Spoofing: attempts to impersonate a principal
- Tampering: unauthorized altering
- Repudiation: denying responsibility for past actions.
- Information disclosure: unauthorized release of data
- Denial of service
- Escalation of privilege: obtaining privileges to access often critical resources

Which of the security properties break for each?

3. Security goals

- What are your security requirements
  - Confidentiality (encryption)
  - Integrity (cryptographic hashes)
  - Authenticity (MAC or keyed hash)
  - Availability (DDoS)
  - Auditability (Blockchain, tamper-proof-logs)
  - Access control
  - Privacy
  - Plausible deniability …

4. Designing systems

- Security via policy
  - Pass a law and make it illegal
- Use cryptography and security primitives
  - Encryption, hashes, VPNs, firewalls
- Make your system resilient to attack
  - Keep updated copies of systems (*hot standby*)
- Detection and recovery
  - Intrusion detection system, Redundancy etc.

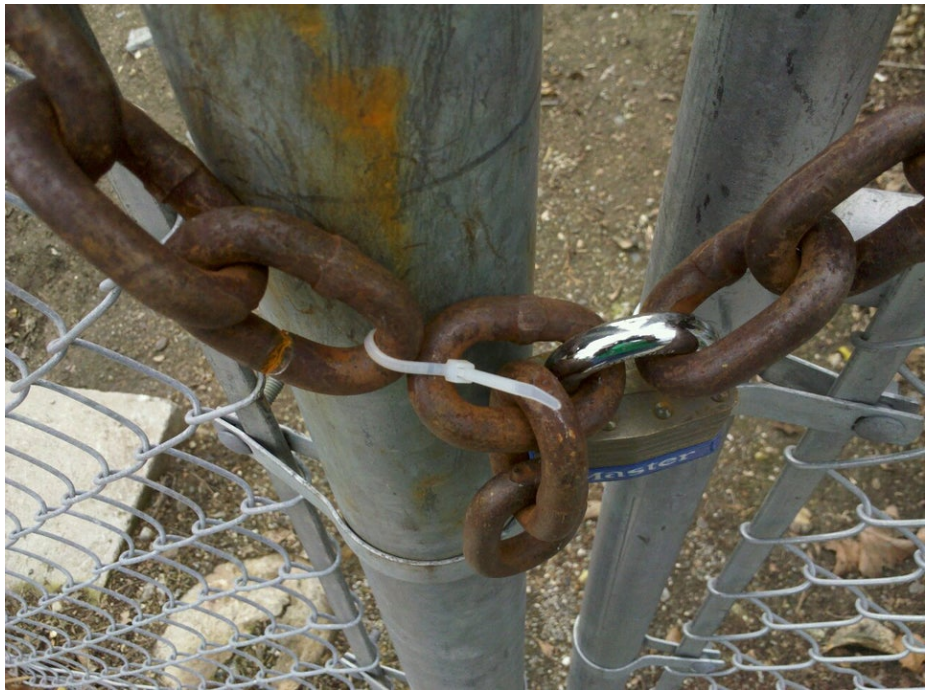
Pitfalls of security

- Can’t protect against everything
  - expensive and inconvenient

- Identify most likely avenues of attack
  - Identify *likely* attackers and their resources?
  - Identify *likely* consequences – financial loss or personal loss?
  - Accept your design will not defend against all attacks
  - Identify where will it not help? (is that reasonable)

You need to think like an attacker

- Adversary target *assets,* not defenses
  - Will try to exploit *weakest part* of the defenses (bribing, social engineering)



Summary

- Security is important AND difficult
- Security is NOT absolute
  - Your solution will depend on YOUR system model, threat/attacker model, security goals
  - Shoot for at least “raising the bar”

- Bonus: System and attack model of obtaining encrypted data
  - Security by obscurity
  - Kerckhoffs’s law/Shannon’s maxim (“Enemy knows the system”)

Roadmap

- Basic security analysis
  - Threat modelling
  - Adversary modelling
- Design principles for security

Some secure system design principles

- Make defaults safe (e.g., use https by default)
- Make the design open: more eyeballs, often better security
- Principle of least privilege: E.g., never use a root account (use “sudo”)
- Least surprise: User mental model should align with your system design
- Never trust the input: Always verify (integrity, authentication etc.)
- Isolation: compartmentalize resources (e.g., hardware isolation, disk partition, virtualization, sandboxing, firewalls)
- KISS (Keep-it-simple-stupid): Keep designs as simple and small as possible (e.g., TPM)
  - minimize the trust base
  - Minimizes attack surface

Some secure system design principles

- Make defaults safe (e.g., use https by default)
- Make the design open: more eyeballs, often better security
- Principle of least privilege: E.g., never use a root account (use “sudo”)
- Least surprise: User mental model should align with your system design
- Never trust the input: Always verify (integrity, authentication etc.)
- Isolation: compartmentalize resources (e.g., hardware isolation, disk partition, virtualization, sandboxing, firewalls)
- KISS (Keep-it-simple-stupid): Keep designs as simple and small as possible (e.g., TPM)
  - minimize the trust base
  - Minimizes attack surface