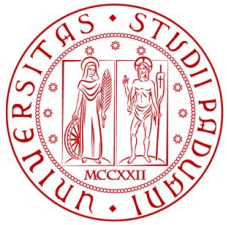
Mobile Security

Dr. Eleonora Losiouk
Department of Mathematics
University of Padua
elosiouk@math.unipd.it
https://www.math.unipd.it/~elosiouk/



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

SPRITZ
SECURITY & PRIVACY
RESEARCH GROUP

DIPARTIMENTO
MATEMATICA

Last class...

- We explored **what an attacker would like to do**...
- **List of possible malicious behaviors** assuming an attacker has arbitrary code execution and arbitrary privileges

This class

- We explore the **"how"** an attacker can do what she wants
- How can an attacker bypass **security protections**?
- This class is all about **security vulnerabilities**

Security Vulnerability

- What is a "security vulnerability" (aka "security bugs")?
- A weakness that allows an attacker to perform actions that
  - 1. were **not meant to be possible**
  - 2. has some **negative security repercussions**

Not all vulns are the same

- Some vulns are much more important than others
- Skill to acquire:
  - Understand how to **"classify" a vuln** and determine its **severity**

Threat Model

Key Concept: Threat Model

- A "threat model" outlines what it is assumed an attacker can do and cannot do
  - *Weaker requirements* are implicitly included
  - *Stronger requirements* are implicitly excluded

Key Concept: Threat Model

- Keeping the threat model in mind is of critical importance when discussing **attacks** and **defense** systems
- Attack X is possible under threat model T
  - Without knowing T, all attacks could be trivial / impossible
- Defense system Y is effective under threat model T
  - Without knowing T, all defense systems could be safe / vulnerable

Threat Modelling for Attacks

- "I can write arbitrary files assuming I have code execution as third-party app"
- "I can write arbitrary files assuming I have root access"

Threat Modelling for Attacks

- Rule of thumb to judge usefulness of a "new" attack:
  - If assuming threat model T, you knew how to achieve goal G already (without using the new info) ⇒ not interesting
- There are exceptions: if the new attack shows a completely new technique, then it could still be interesting!

Threat Modelling for Defense Systems

- App sandbox is useless when attacker is root
- A defense system may not be able to protect from all attackers, but if it *protects from "frequent" threat models*, then it's useful nonetheless

Threat Modelling for Defense Systems

- If defense doesn't *restrict anything an attacker could do without it and in any scenario* ⇒ useless

- Not always true: defense in depth!

Threat Models

- There are many possible threat models, some are useful

Common Threat Models

- Malicious app on victim's device with
  - arbitrary permissions
  - permission X and Y
  - zero permission

- **Type of permission** X (e.g., "normal" vs. "dangerous") is very important for risk assessment
  - How easy is it for an attacker to get permission X?
    - Requirement for user's "permission acceptance" makes everything more complicated

Common Threat Models

- Attacker knows victim's **phone number**
- Attacker can lure the victim to **visit an arbitrary URL**
- Attacker is on the **same "network"** of the victim
- Attacker can run code as a **privileged user** (root, system)
- Attacker can run code in the **kernel**
- Attacker can run code in a **Trusted App** (TEE userspace)
- Attacker can run code as within the **TEE OS**
- Attacker has **physical access** to the device

Common Threat Models: Three classes (ref)

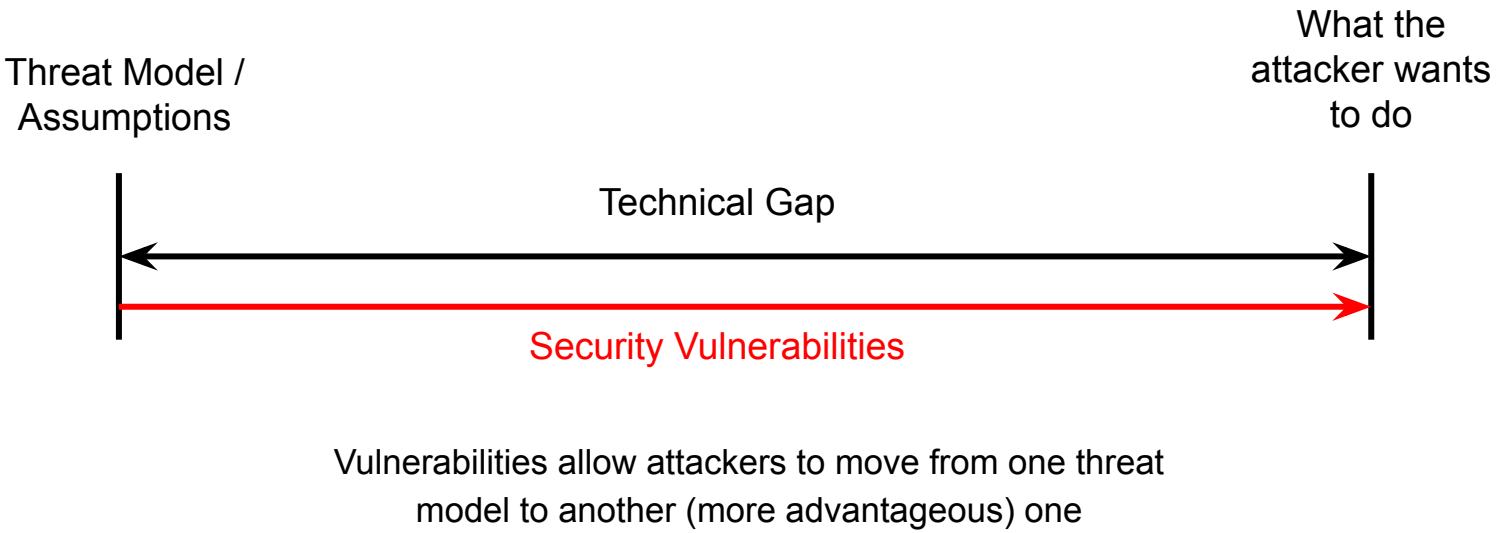
- **Remote** attacker
- **Proximal** attacker
- **Local** attacker

Attack Surface

- The threat model determines the attack surface
- Attack surface
  - "The attack surface is the sum of the different points (the "attack vectors") where an unauthorized user (the "attacker") can try to enter data to or extract data from an environment." (wiki)

Why attackers like vulns?

- There is usually a gap between "threat model" and "what the attacker wants to do"



Vulnerabilities allow attackers to move from one threat model to another (more advantageous) one

Vulnerability Characterization

The "type"

- The "type" aka "what kind of bugs?" aspect
- High-level "type" categories
  - EOP, RCE, DOS, ID, ...

The "type"

- EOP: Elevation of Privilege
- RCE: Remote Code Execution
- ID: Information Disclosure
- DOS: Denial-of-Service

The "where"

- Vulns are more/less important depending on **which device hardware/software component** is affected
  - I.e., the "where is the bug?"

The "where"

- A number of relevant process types (ref)
  - *Constrained* process (process more limited than a normal application)
  - *Unprivileged* process (third-party app)
  - *Privileged* process
  - *Trusted Computing Base* (TCB)
    - kernel, has capability to load scripts into a kernel component, baseband
    - kernel-equivalent user services (init, ueventd, vold)
  - *Bootloader*
  - *Trusted Execution Environment* (TEE)

The "where"

- Two aspects of the "where"
  - In *which component* is the bug?
    - Is this component privileged?
  - In *which part of such component* is the bug?
    - Input processing?
    - Can the attacker reach it?
      - If yes: great!
      - If no: no matter what the bug is, it may be useless

Type + Where

- The combination of the "type" of bug and "where" it is (i.e., which component is affected) determines its severity and relevance

- How "easy" is it to "exploit" and how much "powerful"?

Vulnerability Exploitation

- *Exploitation* is the process of "taking advantage" (i.e., "exploiting") a vulnerability so that an attacker can perform unintended actions
- Specific vulns may be exploited only by specific attackers
- *Threat model* analysis tells us *"which types of attackers can exploit which vulns"*

Vulnerability Severity

- **Severity == "ease of exploitation" x "damage"**
- Ease of exploitation
- Damage of a vuln also depends on the type of victim

Severity Score

- Google assigns a **"severity" score** to each bug
  - Low, Moderate, High, Critical
- The score is assigned depending on the combination of
  - *type of bug* / what the attacker can achieve
  - which *component* is affected
  - under which *condition* it can be exploited
- The score determines **how Google prioritizes its fix and deployment of the patch**

Severity Score

- Many different possibilities
  - Conceptually, a severity can be assigned to each combination of the relevant aspect
  - See Android's full list here
- More generic alternative (but of dubious utility)
  - CCVS: Common Vulnerability Scoring System (wiki)
  - Numeric score from 0 to 10
  - Several metrics are combined via a numeric expression
    - Access vector, attack complexity, authentication, impact vs. CIA triad, ...

Vulnerability Chaining

- Attackers can take different bugs and "chain" them
- Chainspotting: Building Exploit Chains with Logic Bugs
  - Chain of 11 bugs across 6 unique applications
  - Net effect: remote attacker can install + run arbitrary APK

Vulnerability Tracking

- Thousands of bugs are discovered every year
- Important to track them + convenient way to address them
  - To *reference* specific vulns when discussing about their details
  - To *describe* which vulns are fixed in a security update of an affected component
  - To *distinguish between similar* (but different) vulns affecting the same component

Vulnerability Tracking: CVE numbers

- **Common Vulnerabilities and Exposures** (CVE) system provides a reference for known security vulns
- They are in the **CVE-<year>-<id>** format
  - Example: CVE-2015-3864
  - The CVE is assigned during the vuln disclosure / fixing process
- Several databases that systematize available information
  - https://cve.mitre.org

Android Security Bulletins

- You now know everything you need to fully understand Android monthly security bulletins

- CVE, type, component affected, severity (and its rationale), relevant threat models, attack vectors

Questions? Feedback? Suggestions?