THE UNIVERSITY *of* EDINBURGH

# System Security Revision

Security Engineering (Spring 2026)

Lecturer: Jingjie Li & Daniel Woods



## General Tips

- Understanding the systems stack
- Threat modeling and identifying threat surfaces
- Identifying and designing defense techniques and strategies
- Justifying tradeoffs technically and economically, thinking about the stakeholders

# E1

- What makes deperimeterised network more secure? Identify TWO loopholes or potential security vulnerabilities in a deperimetrised university network.



# E1

- What makes deperimeterised network more secure? Identify TWO loopholes or potential security vulnerabilities in a deperimetrised university network.
  - Start from the concept, here you may want to compare deperimeterised network to parimeterised network. Thinking about the threat model
  - Thinking about the context and explain the context. You may want to make hypothesis, e.g., who will get access to the university network?



# E2

- Identify two physical security vulnerabilities of smart home devices. How likely these security vulnerabilities are more likely to be exploited? Who will be the most likely adversary for each vulnerability?



# E2

- Identify two physical security vulnerabilities of smart homes. How likely these security vulnerabilities are more likely to be exploited? Who will be the most likely adversary for each vulnerability?
  - Thinking about both physical and cyber-physical interfaces
  - Be more specific about the context when you want to justify an assumption
  - Thinking about broader mindset in security engineering, in this case, who are the adversaries?



# E3

- How may an attacker exploit power side channel for a smart phone? Propose a measure to mitigate this attack and discuss the cost tradeoffs in implementing it.



# E3

- How may an attacker exploit power side channel for a smart phone? Propose a measure to mitigate this attack and discuss the cost tradeoffs in implementing it.
  - Define power sidechannel, and think about the surfaces to measure power for analysis
  - Based your threat surface and where the adversary is at, think about both digital and physical defense



# E4

- How could a VM hypervisor preserve security? What security limitations a VM hypervisor may have?



# E4

- How could a VM hypervisor preserve security? What security limitations a VM hypervisor may have?
  - Start from the system stack, link to the concept (isolation), explain how isolation works for VM, e.g., what are isolated, and why it's relative to security?
  - We are not asking for full details of the implementation, e.g., how does page-table walk work exactly.
  - Thinking about tradeoffs from different security perspectives, e.g., proactive or reactive, or from technical surfaces, or assumptions of trust



# E5

- Why the app-in-app (miniapp) ecosystem may make security control or assessment more challenging? Why companies and app developers are still interested in developing them?



# E5

- Why the app-in-app (miniapp) ecosystem may make security control or assessment more challenging? Why companies and app developers are still interested in developing them?
  - Thinking about the implementation and the vertical system stack
  - Who are in which layer?
  - Are people's interest aligned?

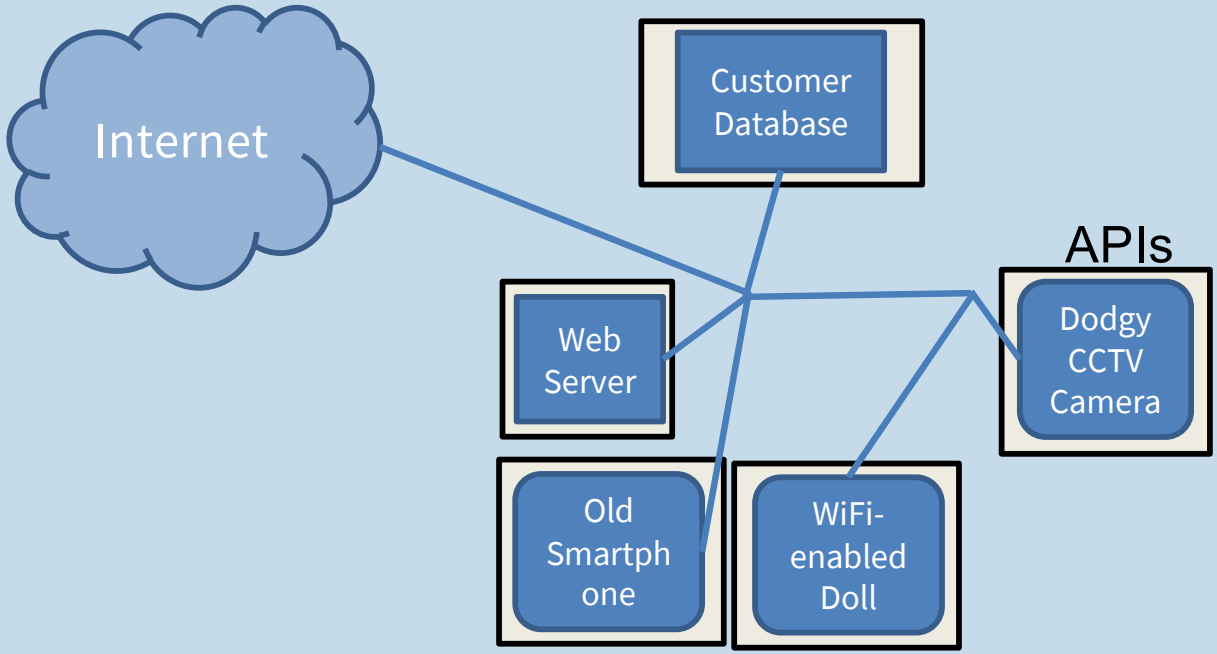


## Topics quick review

- Networking
- Physical
- Hardware
- OS
- Ecosystem

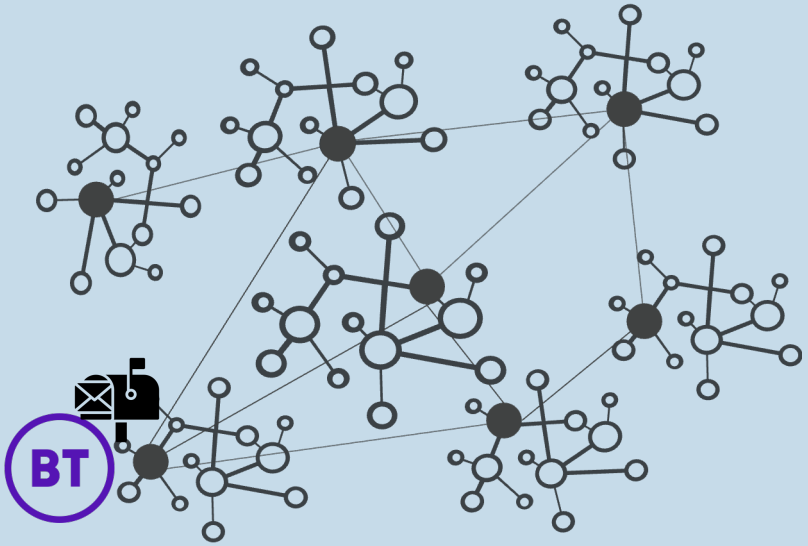
# Deperimeterisation

e.g. Google's BeyondCorp




Shifting access control from network perimeter to individual user and devices

# BGP




- Used for networking between Autonomous Systems in the internet (e.g. ISPs, telcos, large organisations)

# DOS amplifier

Spoofed IP

**"C"** -> B: SYN; my number is X
B -> C: **ACK**; now X+1
SYN; my number is Y
B -> C: **ACK**; now X+1
SYN; my number is Y
B -> C: **ACK**; now X+1
SYN; my number is Y
…

TCP Syn Reflection

**"C"** -> B: SYN; my number is X
B -> C: **ACK**; now X+1
SYN; my number is Y_cookie

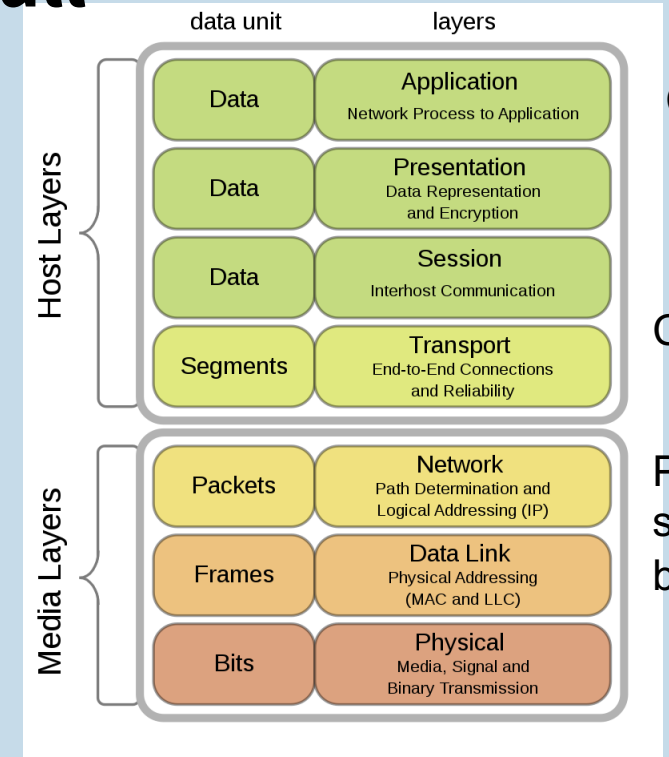
(until a correct copy is received)

Not stored

TCP SYNcookie

- What is the cost of a SYNcookie?

# Firewall

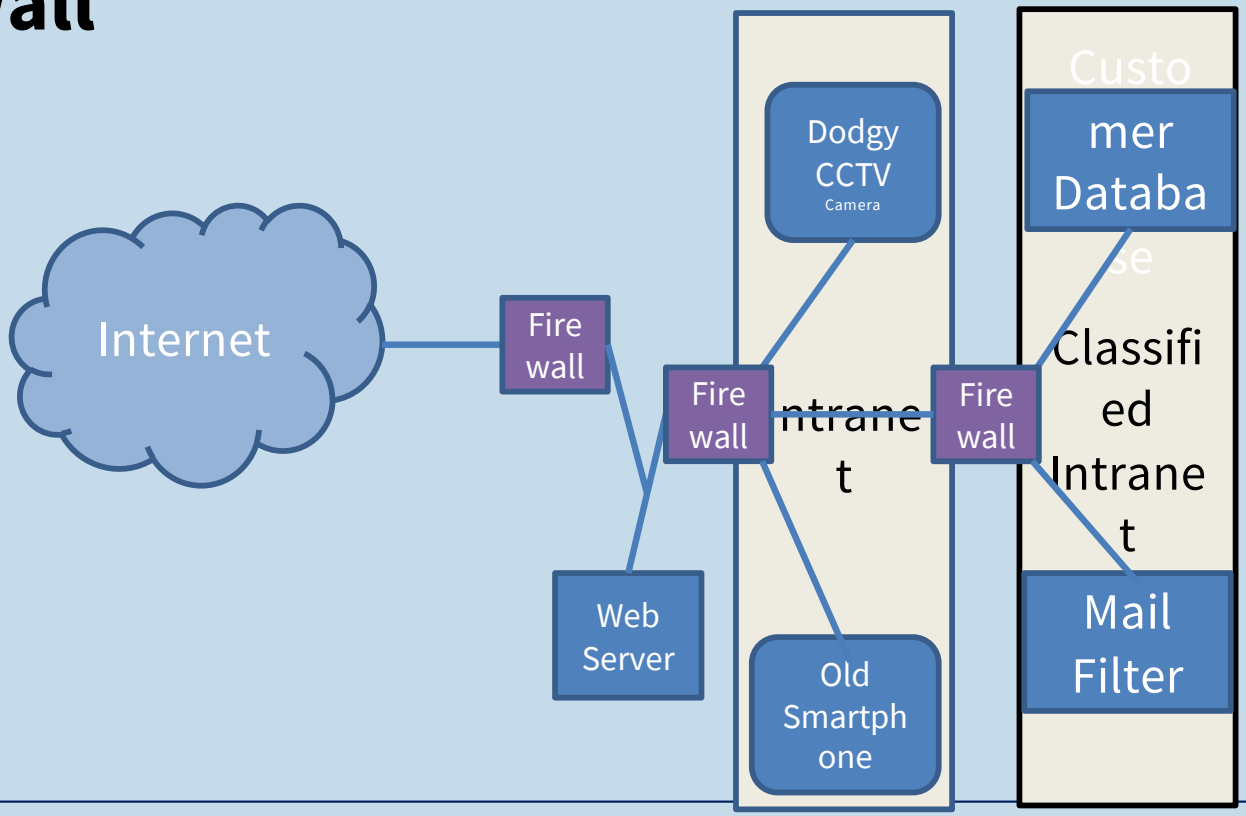



Content Filtering: Application Layer

Circuit Gateways: Full TCP sessions.

Packet Filtering: IP-address spoofing, IP deny lists, port blocking

# Firewall


Internet
Fire wall
Fire wall
Fire wall
Web Server
Dodgy CCTV Camera
Old Smartph one
Custo
mer Databa
Classifi ed Intrane t
Mail Filter



# Email security

- Confidentiality: 95% of emails are with 5 big webmail providers, providing Transport Layer Security
- Unwanted emails
  - Legitimate sources: blocking / marking spams
  - Against DDoS-like campaign
    - Domain keys identification mail, signing the email with key in domains' public record
    - Sender policy framework,  tracing mail to source IP, does not allow forwarding
    - Domain-based Message Authentication, Reporting and Conformance, telling what recipient should do when authentication fails
    - Machine learning based systems

## Physical Security Philosophy

- Locks, and walls, will be some part of your infrastructure at some level
- While the techniques are simpler than digital security, the weaknesses are often as subtle.
- **Five stage of physical security: Deter-detect-alarm-delay-respond**
  - **Time matters!**



## Attacker Capabilities in Threat and Risk Assessment

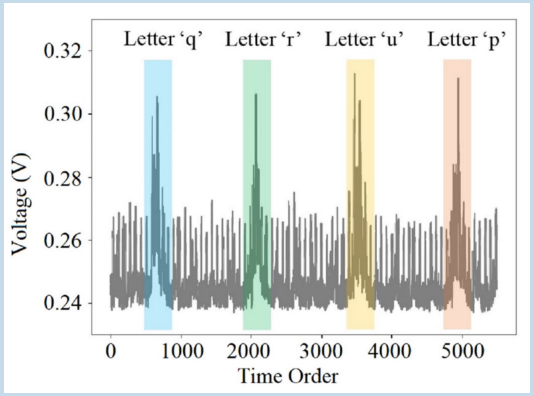
- Derek – 19-year old addict, **opportunistic** criminal looking for simple low-risk opportunities
- Charlie – 40-year old with 7 convictions, Not intelligent, **but cunning and experienced**, so knows the tools of the trade
- Bruno – "**gentleman criminal**" who steals art and takes pride in his work. Bruno is adept at lock and alarm hacking, and is interested in getting into computer hacking too.
- Abdurrahman – head of a dozen agents. He has access to **specialist weapons and PhD-grade technical support**
- Unskilled -> Skilled -> Highly Skilled with help -> Highly Skilled with resources

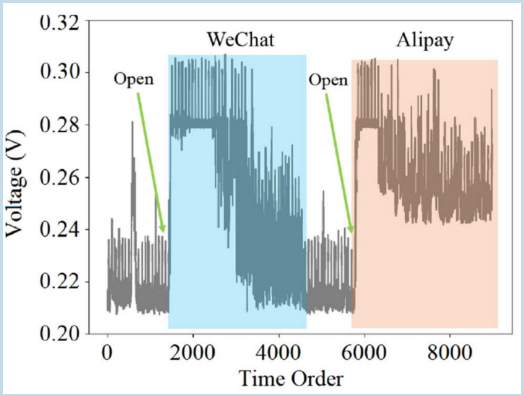
## Temper resistance – Hardware Security Modules

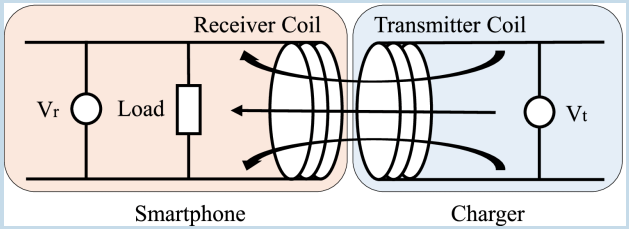
- Store confidential data and perform critical computation
- RAM set to 0 (destroying and refreshing encryption keys) when the physical case is open
- Meaning maintenance people can't get the key
- Early version vulnerable to cut through and people "seal" cores with epoxy resin
- Still leave information somewhere?

# Side channels: physical channels carry more information than you want

- Information breach from side channels

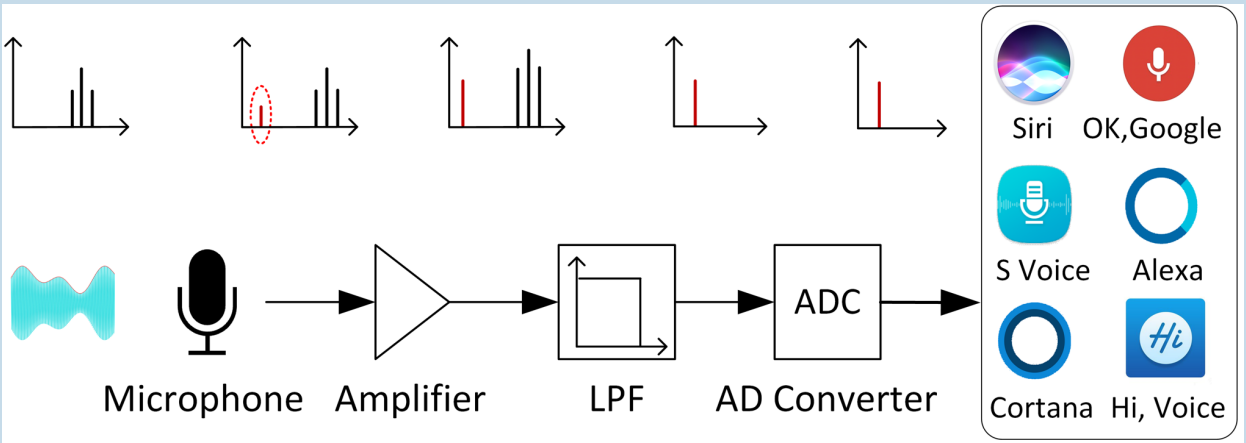


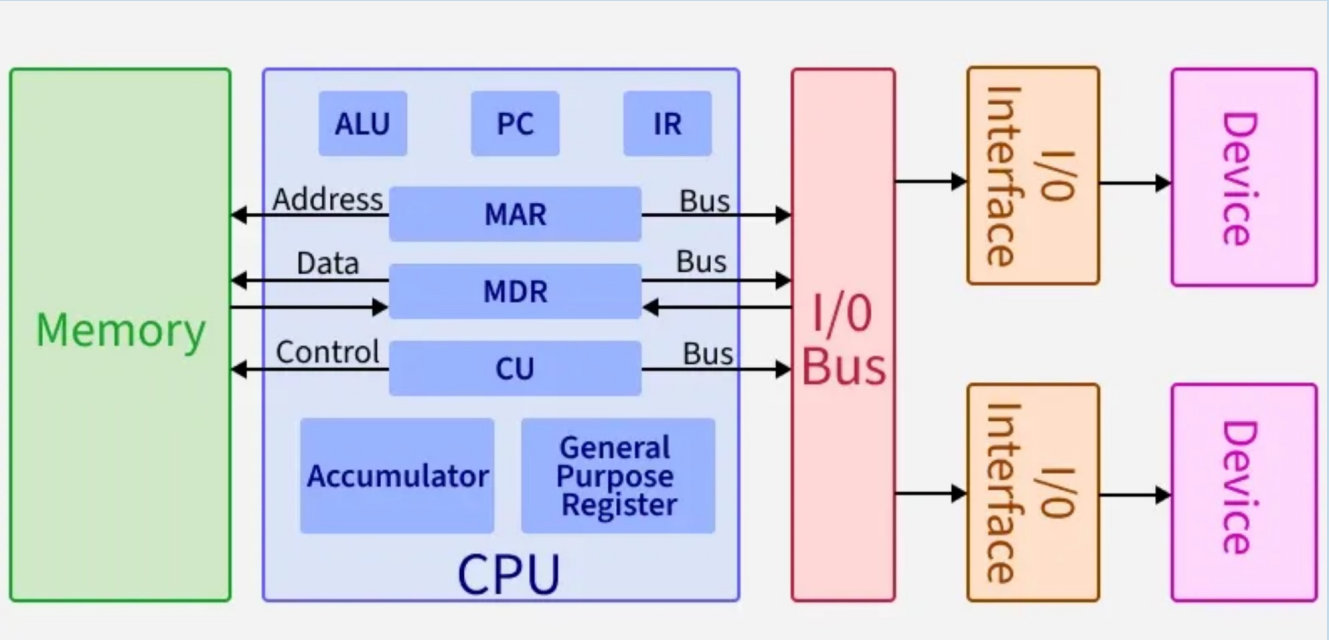

Liu, J., Zou, X., Zhao, L., Tao, Y., Hu, S., Han, J. and Ren, K., 2022. Privacy leakage in wireless charging. *IEEE Transactions on Dependable and Secure Computing*, *21*(2), pp.501-514.

# Covert channels: leveraging physical "fault" to inject malicious input



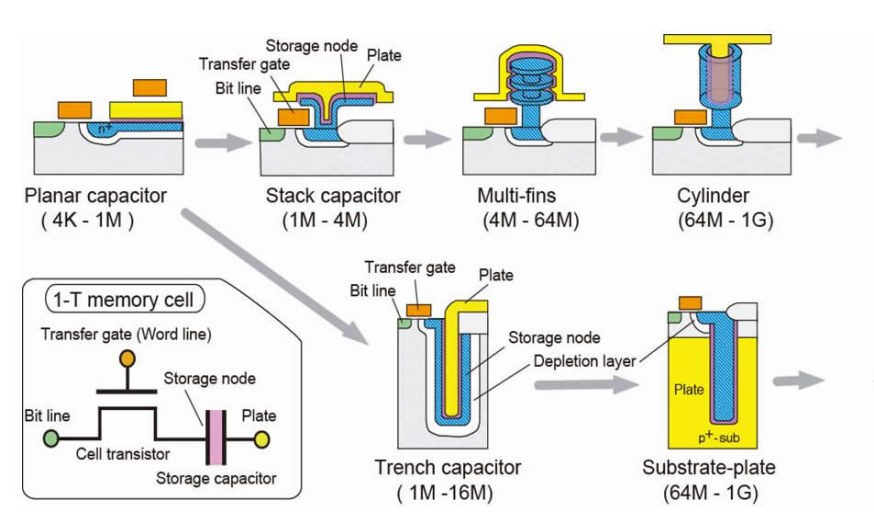

- How to inject malicious comment without human notice?

# CPU architecture



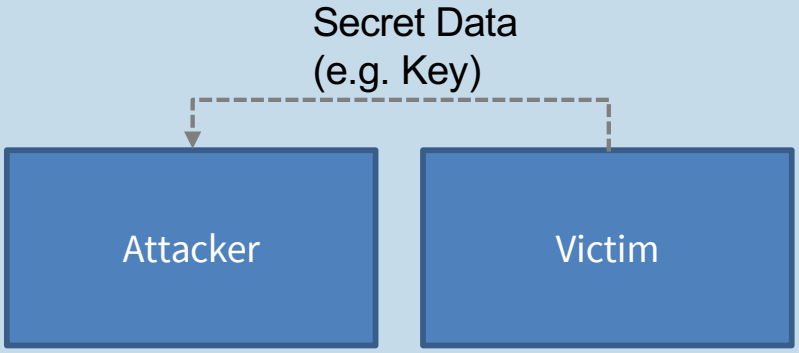

https://www.geeksforgeeks.org/computer-organization-architecture/computer-organization-von-neumann-architecture/

# Hardware architecture side channel - Rowhammer

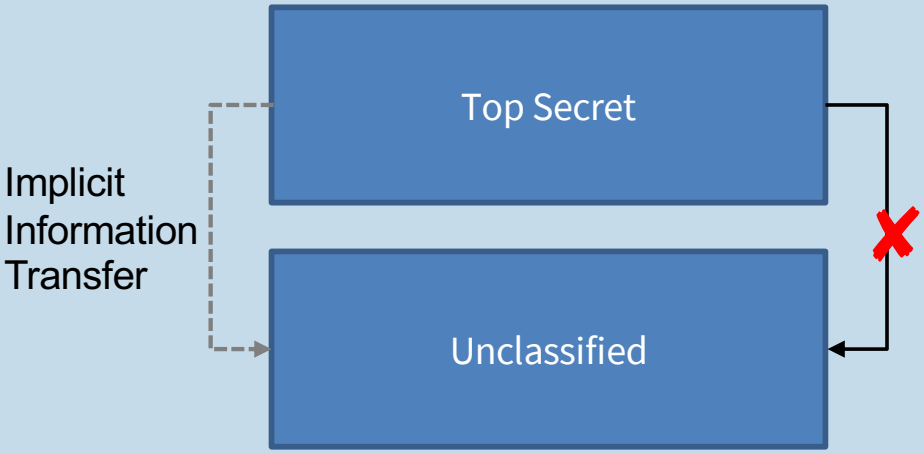



- Bits stored in capacitors, e.g., 1 = charged, 0 = discharged
- It leaks overtime! (e.g., 0 -> 1)
- Charge needs to be refreshed to keep data

## Channels

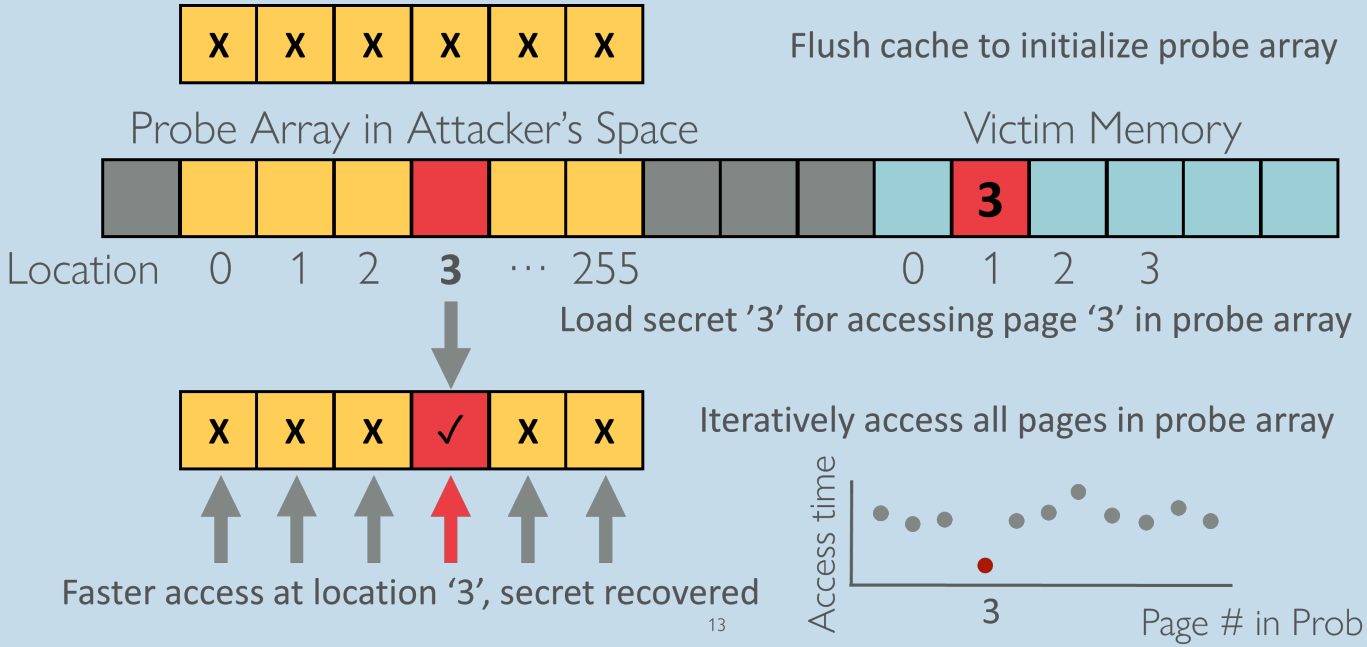



Side channel




Covert channel (when the leak of information is deliberate)

# Meltdown – covert channel: Flush + Reload

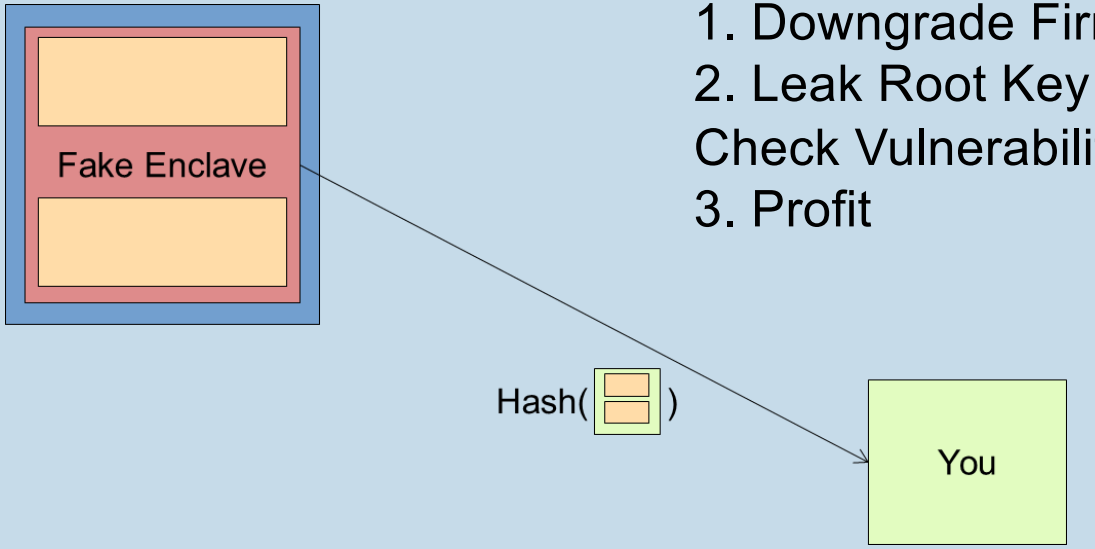




## Meltdown – Countermeasure

- Hardware
  - Disable out-of-order execution
  - Serialize permission check and register fetch
  - Hard-split user and kernel spaces
- Software: KAISER (KPTI)
  - Only some privileged memory mapped in user space for switching to kernel mode
  - Kernel protected from being accessed from user space in kernel mode

## Enclave – am I really running things in an enclave?




1. Downgrade Firmware
2. Leak Root Key through Signature Check Vulnerability
3. Profit

- Enclave security relying on remote attestation. But what get attested?

## Discretionary Access Control - Access Control List

| Subjects (users) \ Objects (files) | a | b | c | d | e |
|---|---|---|---|---|---|
| **jingjie** | r,w | - | r,w, own | - | r |
| **bob** | - | - | r | r | r,w |
| **alice** | w, own | r | r | - | - |
| **eve** | r | r,w | r,w | - | r |

- Access Control Lists: store permissions with file. May need different permissions for different programs, so actually a (user, file, program) triple
- ACLs scale badly without Role-based access control.
- Finding all the files a user has access to is a massive pain.

# Discretionary Access Control - Capabilities

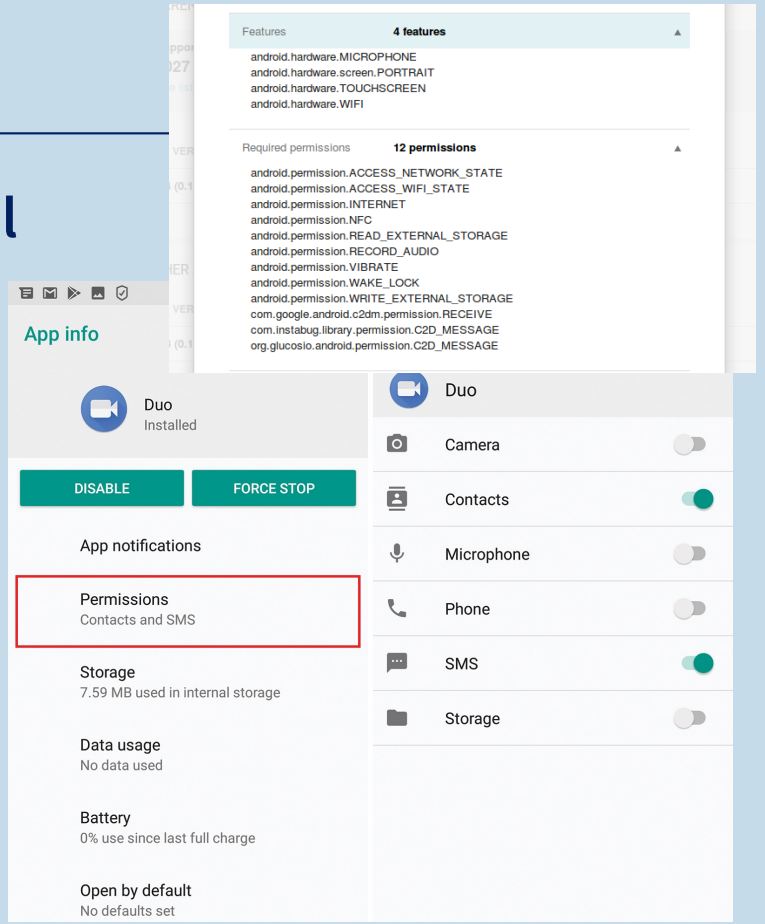
**Objects (files)**

| Subjects (users) | a | b | c | d | e |
|---|---|---|---|---|---|
| **jingjie** | r,w | - | r,w, own | - | r |
| **bob** | - | - | r | r | r,w |
| **alice** | w, own | r | r | - | - |
| **eve** | r | r,w | r,w | - | r |

- Capabilities: store per user, not per file.
- Finding all the users who have access to a file is a pain.
- Hard to revoke access to a particular file, or produce evidence of who could have broken said file.
- Easily transferred
- Public key certificates are really capabilities.

# Case study: Android Discretionary Access Control

- App Isolation: Treat Apps by different companies as different users, using SETUID.
- Permissions also effectively capabilities, implemented by adding GIDs to the list of groups of the SETUID. “Permissions manifests” basically compile down to this.
- Early versions: all granted at install time. So flashlight apps started demanding your address book at install time so they could sell it.
- Since Android 6, Google moved to Apple model of TOFU, but earlier apps still demand on installation.



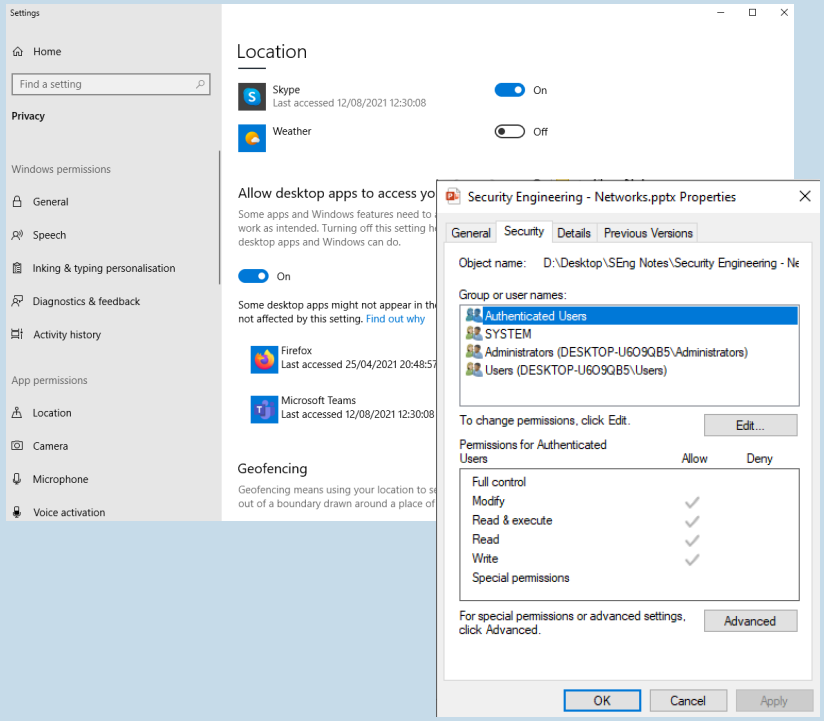


## Case study: Android permissions issues

- API has poor documentation, and the permissions system is often the enemy of the developer, who ends up requesting more permission than they really need.
- Android still has malware! e.g. Pegasus via zero day, but costs $1 million. Alternative markets out of Google's control
- And lots of unpatched devices. The OS-update ecosystem is a disaster…
- Getting access control right intersects with lots of awkward edge cases, e.g. factory reset
- Over-privilege and coarse permissions
- Third-party SDKs inherits the app’s granted permissions.

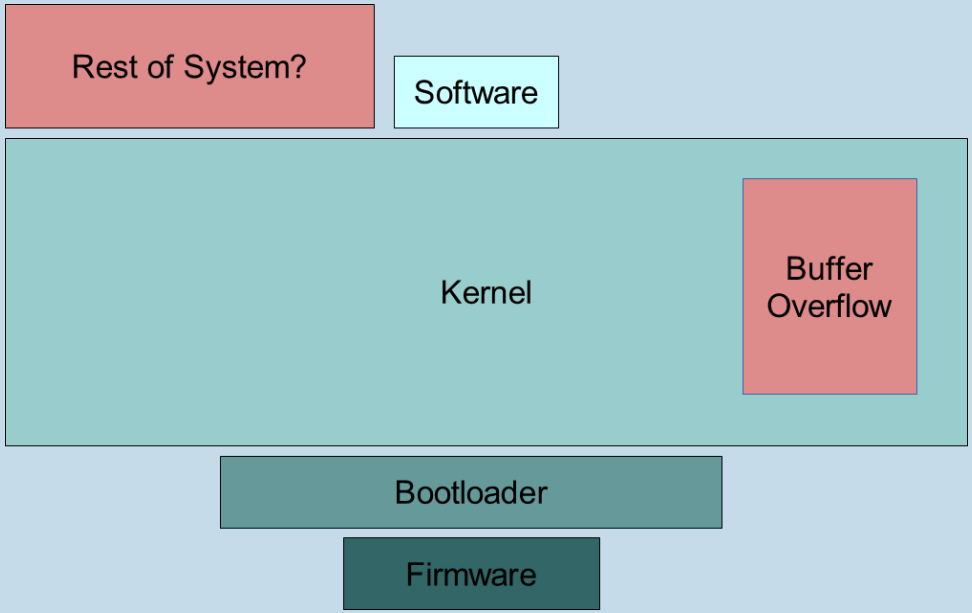
# Case study: Windows – why so complicated?

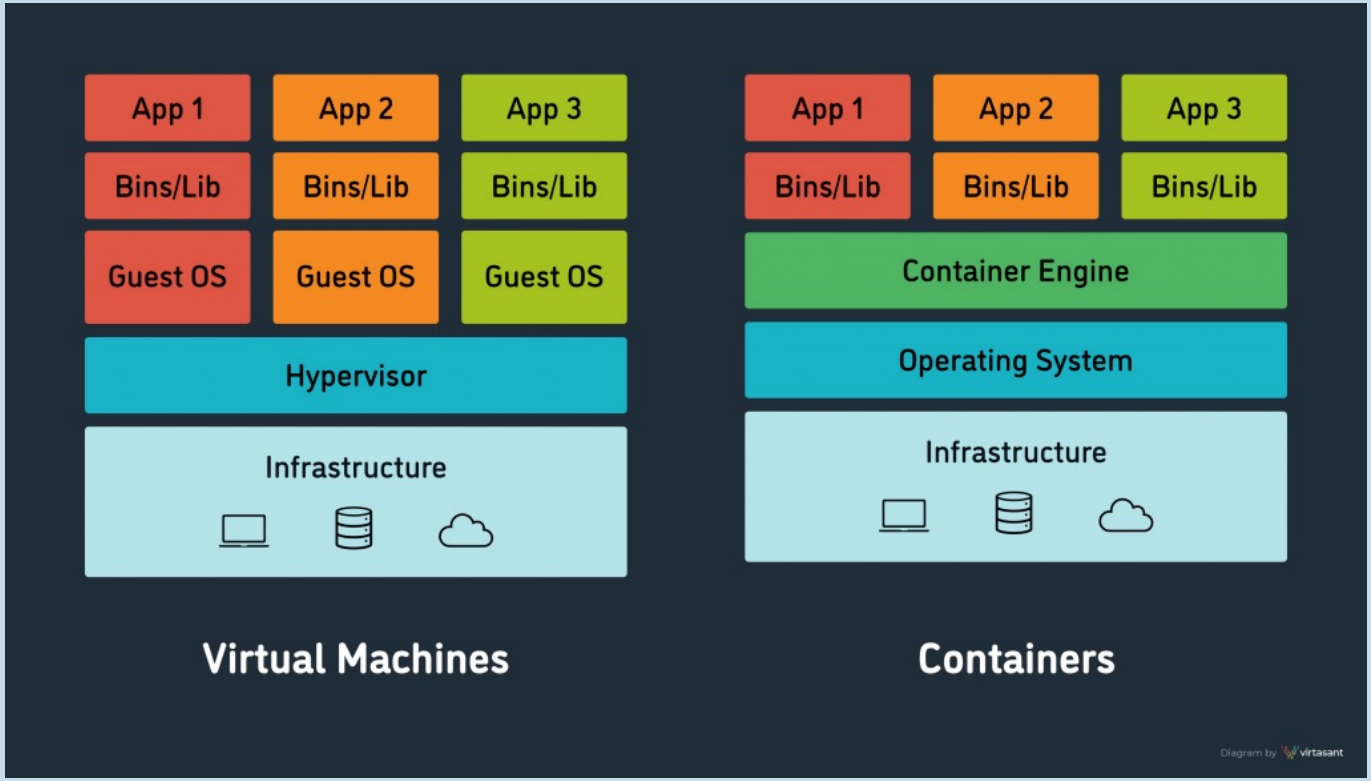
- Corporate customers need complicated access controls. MS made half its revenue from firms >25000 seats.
- Decades of backwards compatibility means testing at scale. And introducing features slowly, and complex compatibility layers e.g. Application Information Service



# Isolation

- How do you stop others, using the same system(s), from being able to read your data / hack your software?
- I mean, really stop them (side channels, bugs in trust computing base)?




App 1
App 2
App 3
Bins/Lib
Bins/Lib
Bins/Lib
Guest OS
Guest OS
Guest OS
Hypervisor
Infrastructure
Virtual Machines
App 1
App 2
App 3
Bins/Lib
Bins/Lib
Bins/Lib
Container Engine
Operating System
Infrastructure
Containers
Diagram by virtasant



## Isolation: Virtualization

- Replaces the entire operating system, and runs a “guest” operating system on top of a “host” via hypervisor.
- Powers cloud computing.
- HW support such as Intel VT-x makes things cleaner and faster.
- Why more secure? The hypervisor can be much smaller than a full OS and so easier to code-review and secure, right???
  - Why hardware compartmentalisation is still useful?
  - VM escape, cross-VM leaks…



## Isolation: Virtualization challenges

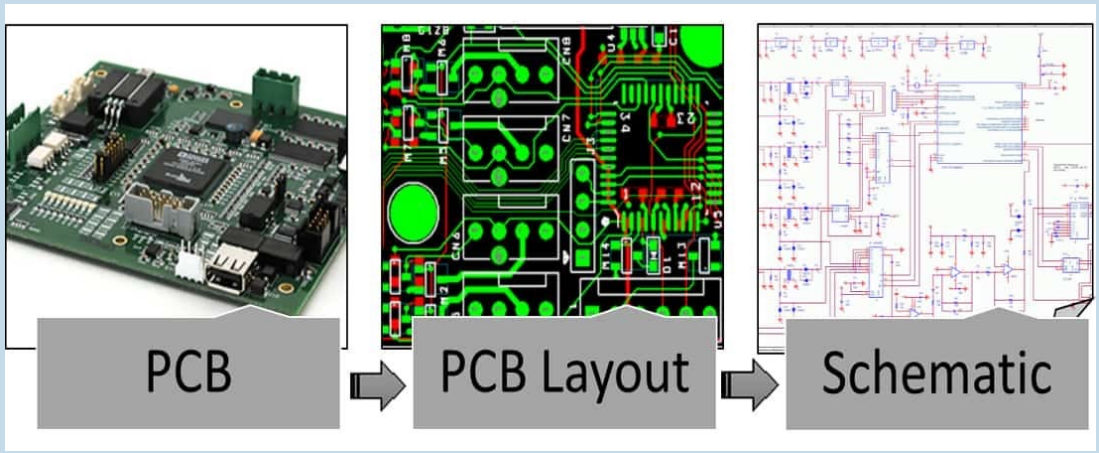
- Subtle issues around monitoring: If you're going to check all ACLs on your server, what about the containers or virtualized systems?
- Trouble at the interface: people still need to share data between VMs and ad-hoc mechanisms such as USB devices
- Bromium: VM per app, messy at the interface with untrusted files sent via host. Need specific exceptions and plugins, like Outlook being prevented from rendering files itself.

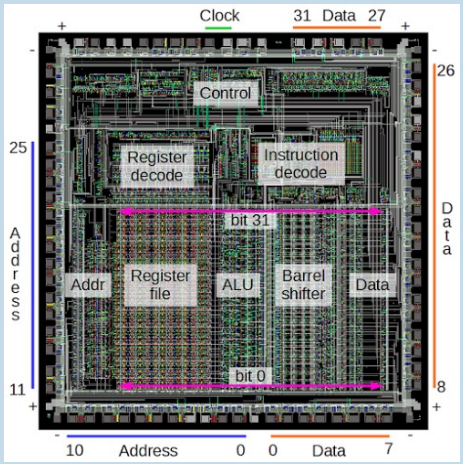


## Isolation: Container challenges

- Not the same as VM, and not really meant for data isolation -- don't expect the same isolation as with a VM -- the trusted code base is still massive.
- Really for deployability, not security. Many subtle bugs, such as blank root passwords as defaults!
- On the flipside, deployability might *be* a security feature – why?

## Hardware supplychain



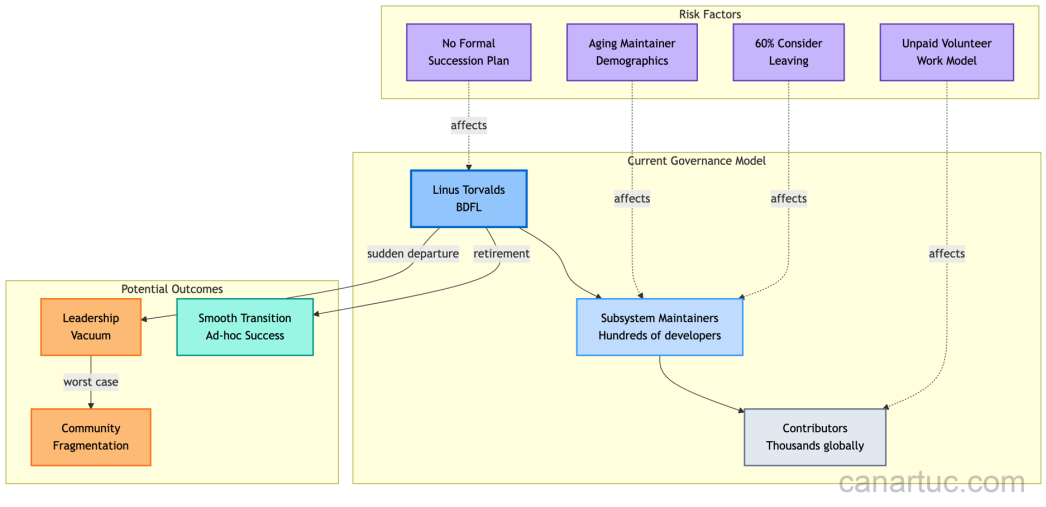




- Designers do not have full visibility into the manufacturing process
- Complexity of the hardware makes bugs and vulnerabilities even hard to find
- Designers/vendors even rely on reverse engineering to ensure IP/chip integrity

# Accessory control


- Locking users and developers through digital right management (DRM)
- DRM via security printing, obfuscation, temper-resistance, license check, etc.

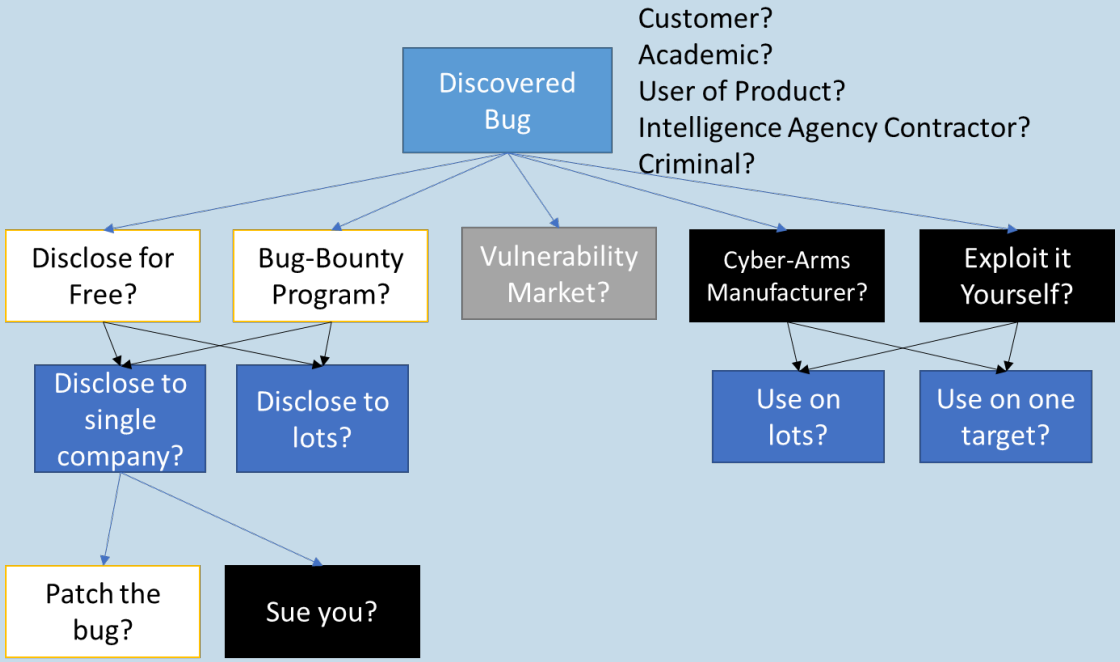
## Software and code supply chain




https://canartuc.medium.com/linux-kernel-maintainer-succession-the-crisis-hiding-in-plain-sight-295105d236b1

- Can insiders (un)intentionally get bad code committed to release?
- Who is an “insider” for the software running in your device? Think about your OS kernel, libraries…
- Code reviews are a form of multi-party authorisation, but be careful to avoid rubberstamping…
- In this scenario, bugs aren’t random – they’re introduced to open-source projects with wide use!

# How to report vulnerability? Or should people even report?




- (Properly) reporting vulnerability is not easy, e.g., avoid zero days
- Risks on the reporters
- Misaligned incentives
- How people benefit from reporting?

# Android platform

| | |
|---|---|
| Mobile Network Operator (MNO) | EE |
| Handset Original Equipment Manufacturer (OEM) | HTC |
| OS Vendor | Google |
| Chip Maker | TSMC |
| Chip Designer | Qualcomm & ARM |

- Google taking control of the Android ecosystem, but still a lot of vendors!
- Updates propagates from bottom
- Who is incentivized to fix a bug? Most likely Google?

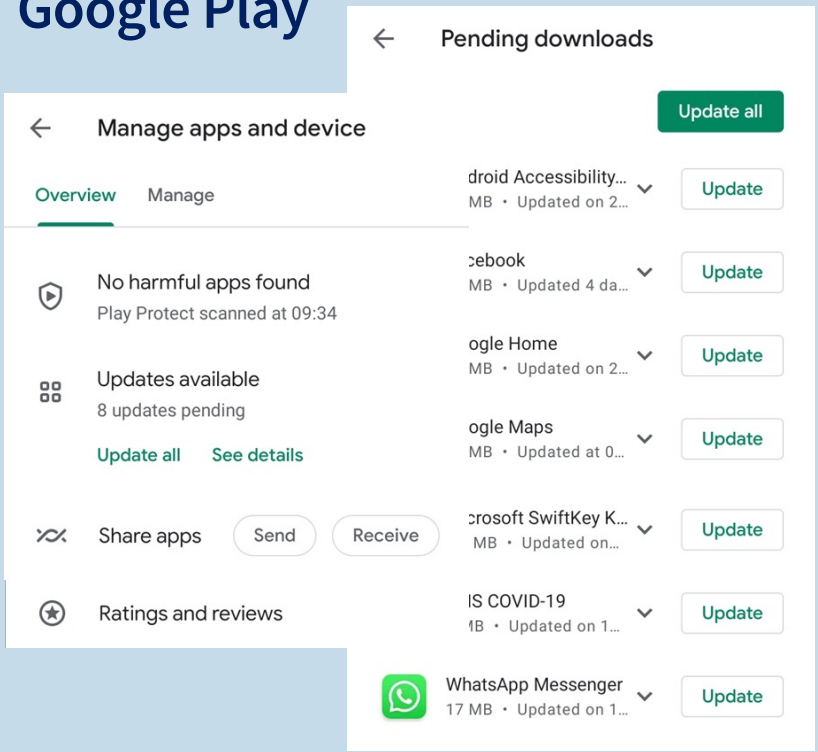
## Appstores

amazon appstore

EXPLORE IT ON AppGallery

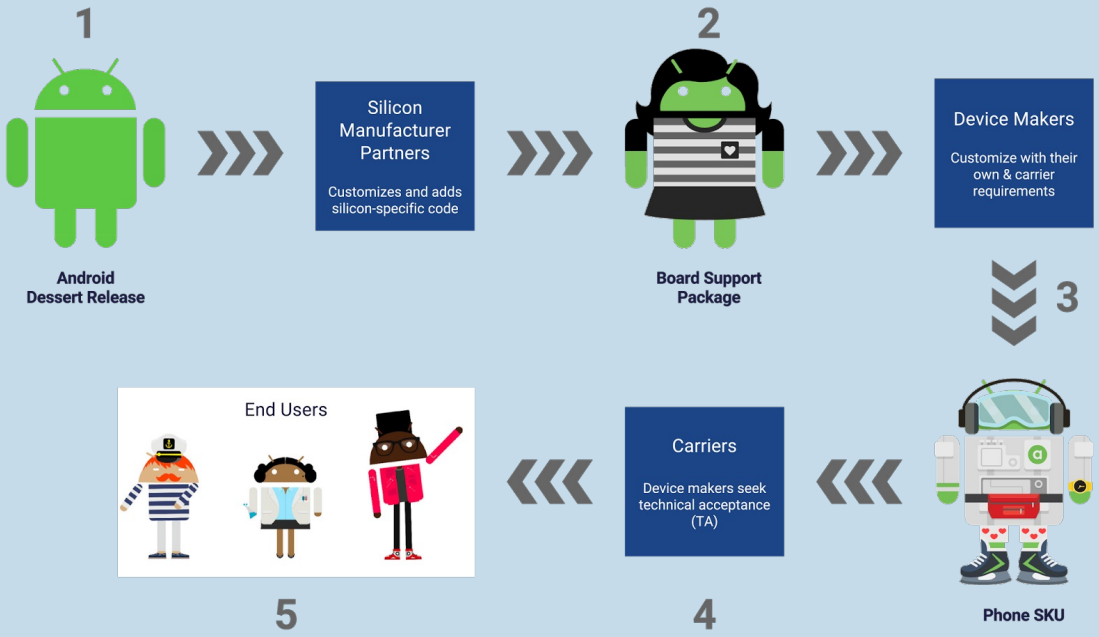
- ~30% of sales profit shared!
- Bundled and locked in entry point – leaving very few options for app developer
- Making apps more predatory (ad, paying, privacy / identity breach…)

## Google Play

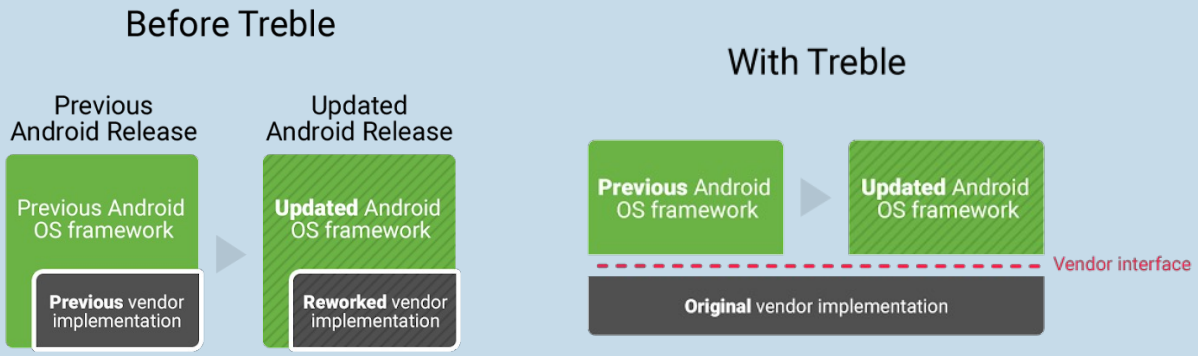



- Self-signed applications (unlike iOS)
- Default with no “Install Apps from External Sources” – security and lock-in
- App Security Improvement Program scanning whole store for harmful apps
- Suite of Sanitizers for User Code: BoundsSan, AddrSan, IntSan, Shadow Stack, Scudo
- Protecting third-party as well, why?

# Android update

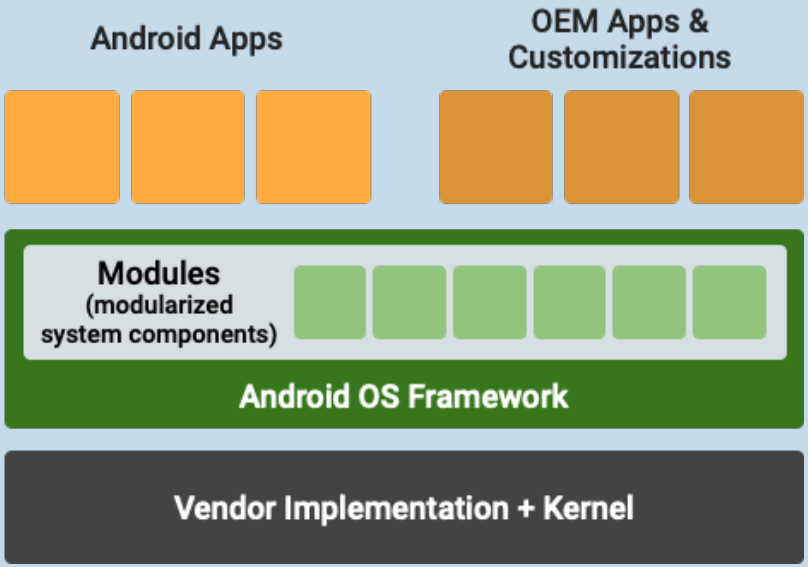


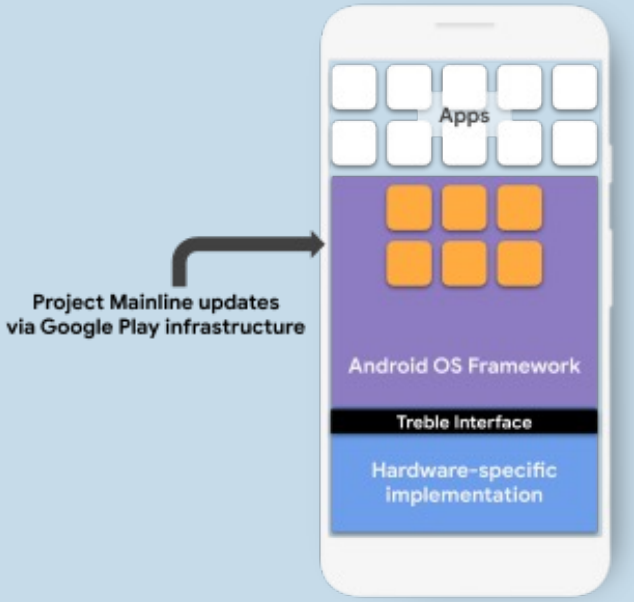
# Android update




- Making OS implementation modular (isolated)
- Updates still need to push through (OEM) original equipment manufacturer…….

# Android update: Mainline




- Further modularization, delivering OS updates through Google Play infrastructure