Security Lancaster | Lancaster University

Secure Federated Learning

Dr. Zhengxin Yu

Senior Research Associate

School of Computing and Communications
Lancaster University, U.K.
z.yu8@Lancaster.ac.uk

Outline

Security Lancaster | Lancaster University

- **Federated Leaning**
  - Background and Motivation
  - Fundamentals of Federated Learning
- **Use Cases of Federated Learning**
  - Federated Learning in Google Products
- **Cutting edge research in Federated Learning**
  - An Adaptive and Robust Federated Learning Framework
- **Open Problems in Federated Learning**

Part 1: Federated Learning

- Background and Motivation
- Fundamentals of Federated Learning

Data Is Available Everywhere

Security Lancaster | Lancaster University

- **Data at the edge**
  - Enable smarter models and better products
  - Generate varied data distributions

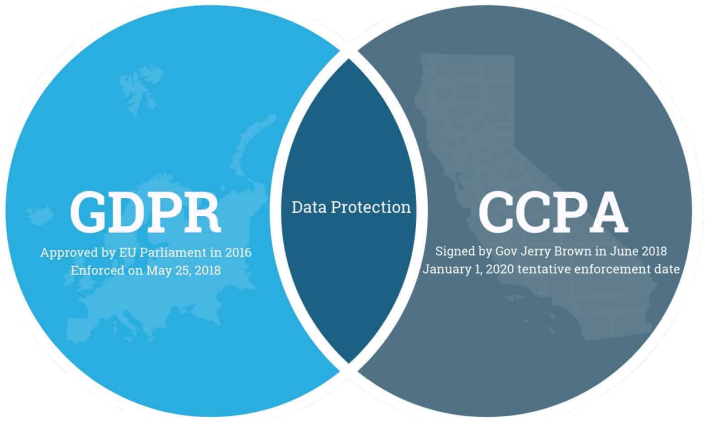
    → highly heterogeneous => harder to process data
- **Data processing is moving on edge**
  - Reduce user latency, work offline
  - Deal with Non-IID, Non-balanced data → some nodes are doing more processing than others
  - Per device: Overfitting, biases → data size

    ← training locally
- **Data exists within isolated silos**
  - Commercially valuable → companies dont want to share
  - Data privacy and security risks

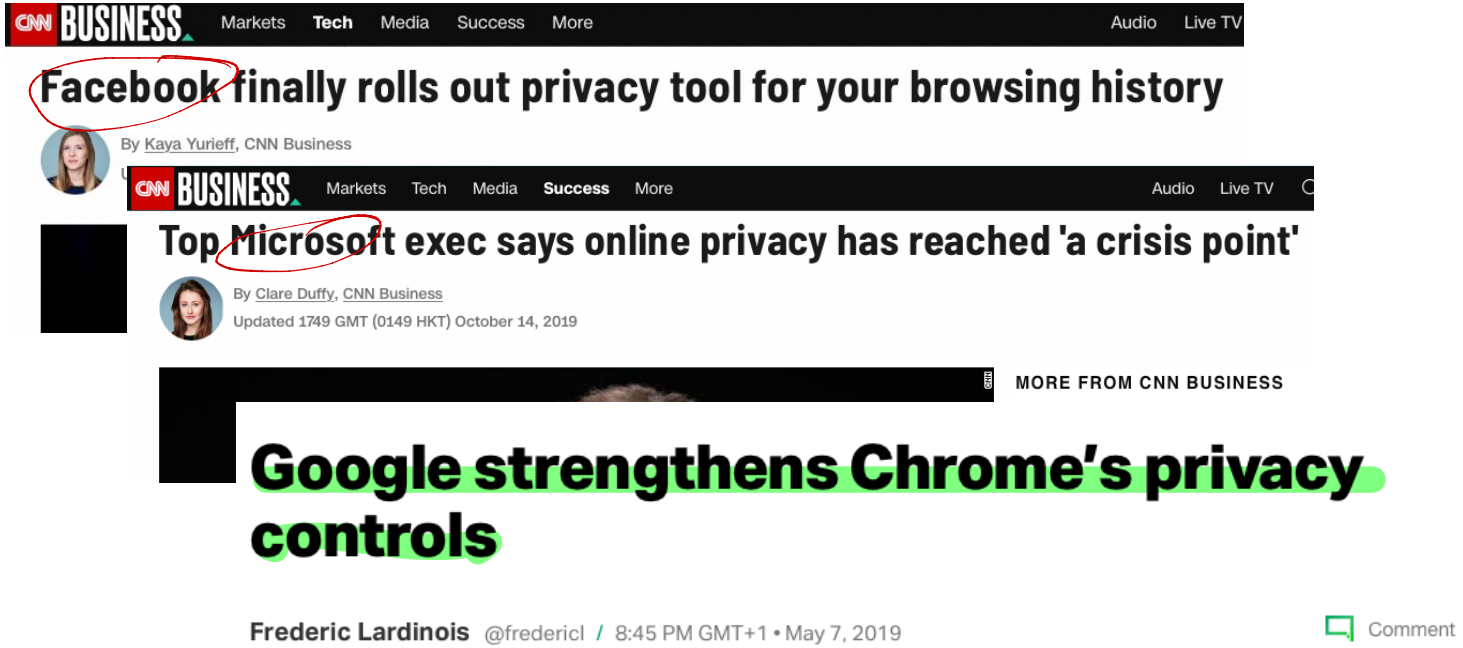
"Data is a precious thing and will last longer than the systems themselves." -- Tim Berners-Lee

Data Protection

- Data growth a concern on user privacy and data security
- Increasingly strict laws on data protection



EU USA

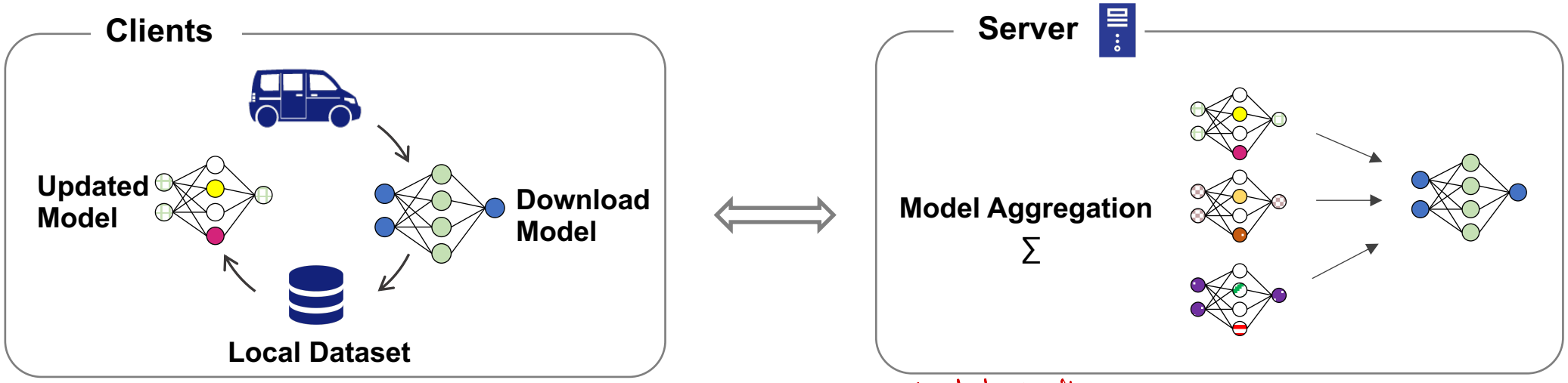


ML Solution: Federated Learning (FL)

Security Lancaster | Lancaster University

Decentralized ML Paradigm: FL

- Collaboratively train **a global ML model** with multiple **clients**
- Keep data decentralized (share models, not data) → reduces security risk
- Data is usable, but not visible



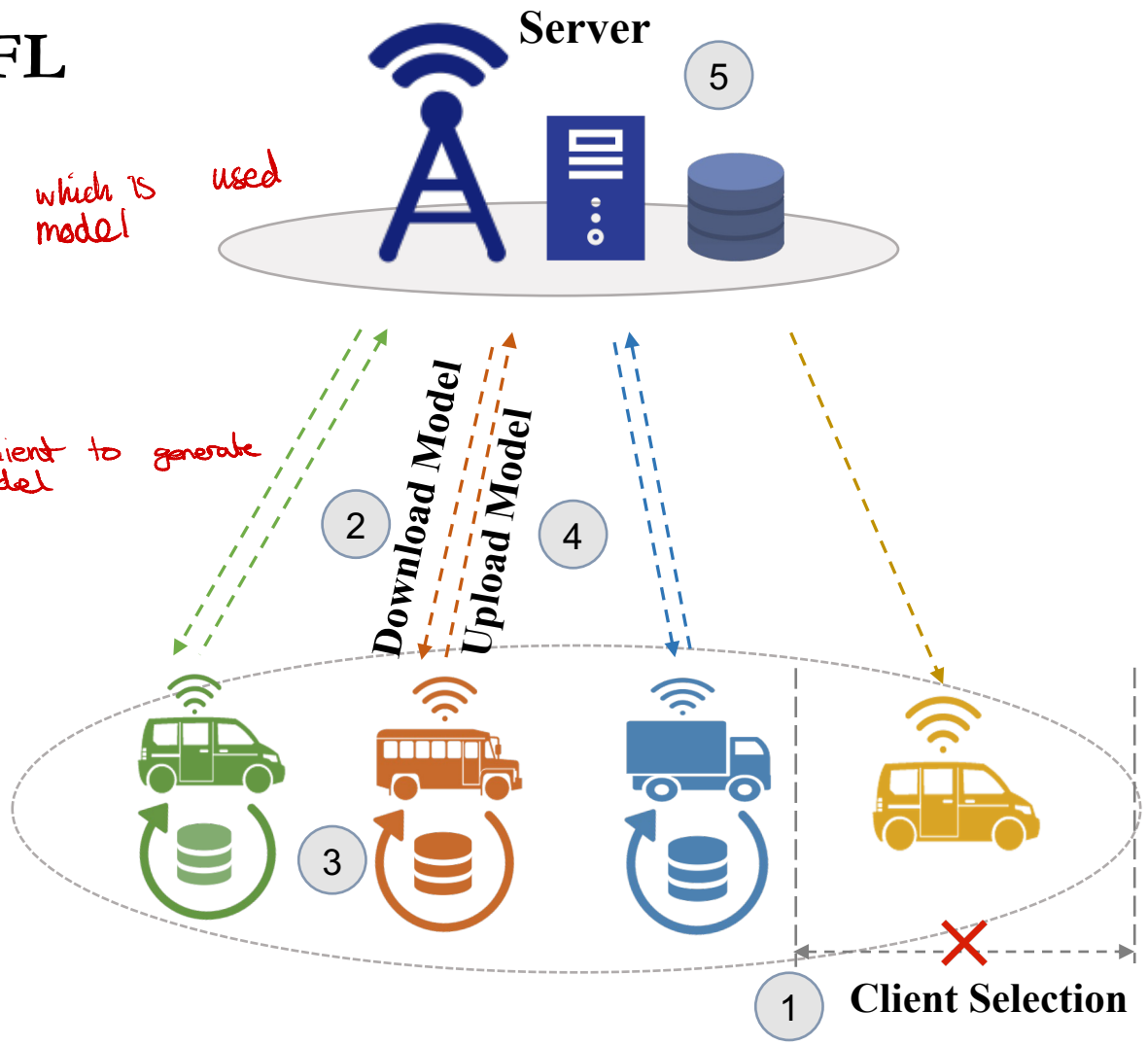
- Clients use their own data to train the model, which is then uploaded to the server
- server aggregates a new model

FL Training Process

Security Lancaster | Lancaster University

One communication round in FL

- Clients selection – only some will attend
- Download model – could be any model which is used to initialise the local model
- Local training
- Upload model
- Model aggregation → Updated from the client to generate a new shared model



Above steps are repeated until an optimal model achieved at the central server

Federated Optimization

Security Lancaster | Lancaster University

- **Objective:**

Difference between predicted & actual values

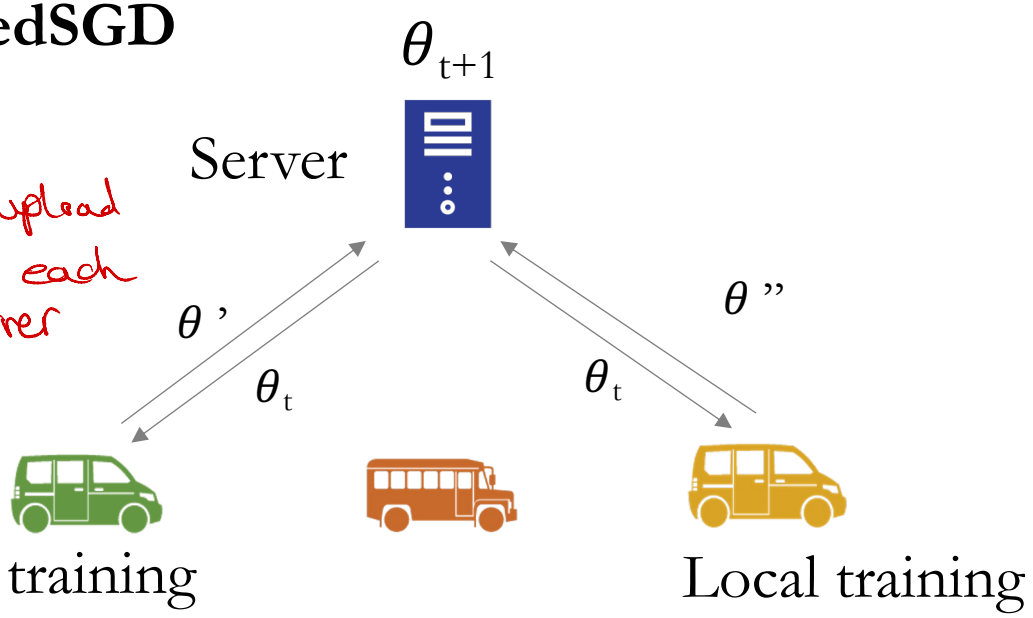
Loss on client k

$$\min_{w} f(w) = \sum_{k=1}^{k} \frac{n_k}{n} \boxed{Fk(\theta)}$$

minimise this

- **Stochastic Gradient Descent (SGD)**
- **FederatedSGD**

To compute and upload the gradients for each client to the server



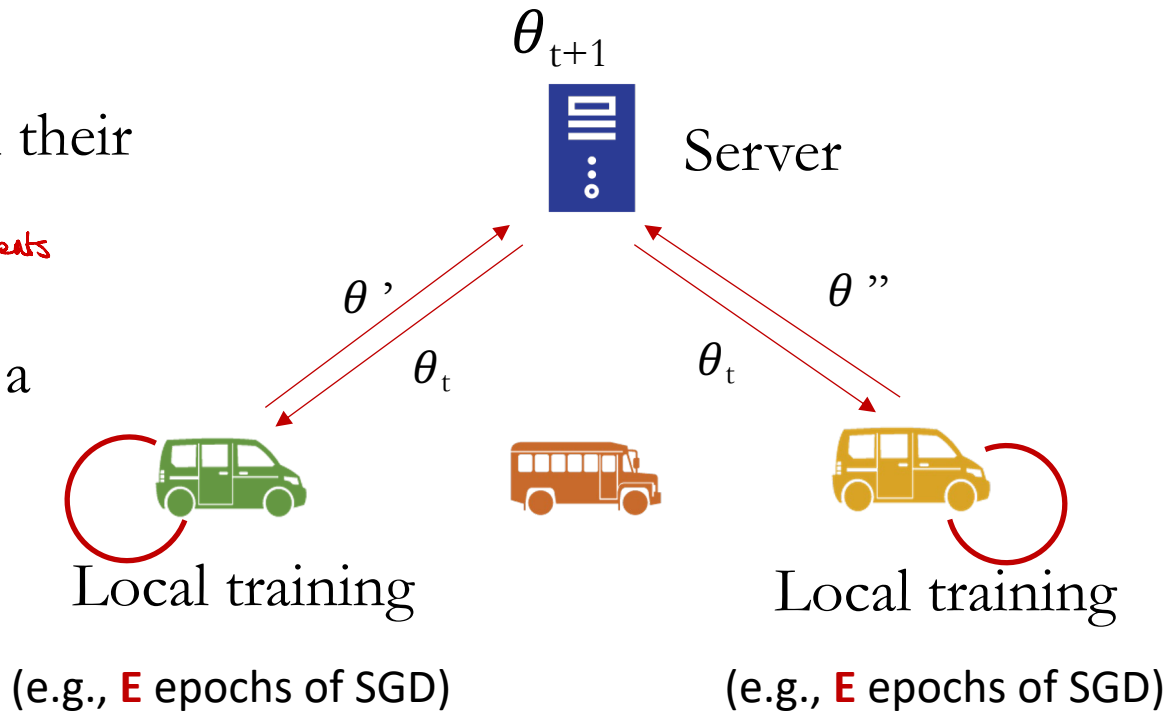
- **Challenge: Expensive communication: massive, slow networks**
  - Can reduce communication in federated optimization by:
    1. Increasing parallelism (more clients)
    2. Increasing computation on each client

⇩

Federated Averaging (FedAvg)

FedAvg

- **FedAvg is a generalization of FedSGD**
- **At each communication round:**
  - Clients run multiple steps of SGD on their local data to compute an update
    From the same initialisation, averaging weights is equivalent to averaging gradients
  - Server computes overall update using a simple weighted average



Distributed Learning vs. FL

Security Lancaster | Lancaster University

Parallising computational power

Training heterogeneous data sizes

	Distributed Learning	Federated Learning
Distribution scale	Single datacenter , 1-1000 clients	Massively parallel, up to 10^{10} clients
Data Distribution	System-controlled(e.g., shuffled, balanced) Any client can read any part of the dataset.	Data is generated locally and remains decentralized. Each client stores its own data and cannot read the data of other clients.
Primary Bottleneck	Computation is more often the bottleneck in the datacentre. GPU is no used to reduce the costs	Communication is often the primary bottleneck, though it depends on the task. → large communication costs
Addressability	Each client has an identity or name that allows the system to access it specifically.	Clients anonymous and interchangeable and cannot be indexed directly
Client Reliability	Relatively few failures	Highly unreliable - 10% or more of the clients participating in a round of computation are expected to fail or drop out

They depend on less powerful communication media or battery-powered devices

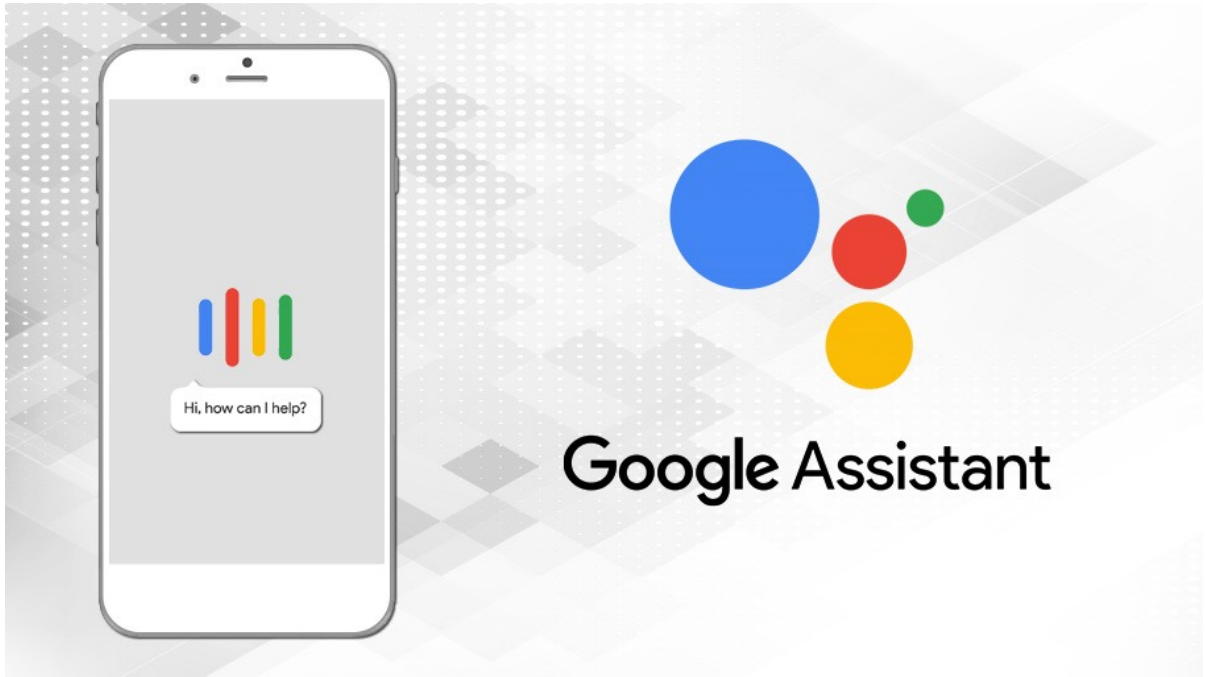
Part 2: Use Cases of FL

- Federated Learning in Google Products

Google Assistant

Security Lancaster | Lancaster University

- FL improves Assistant's "Hey Google" accuracy



- recording audios for updating the model for its triggering logic

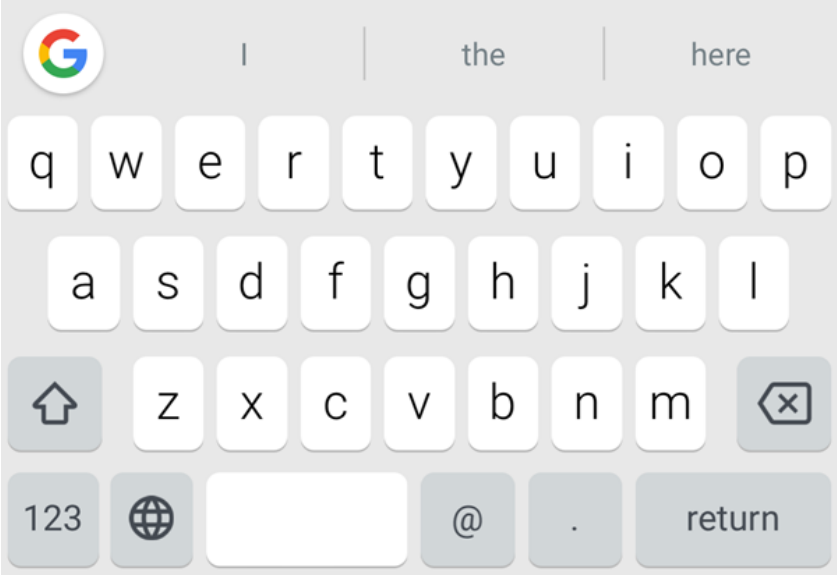
- Upto 20 recordings saved a day, based on quality of interaction

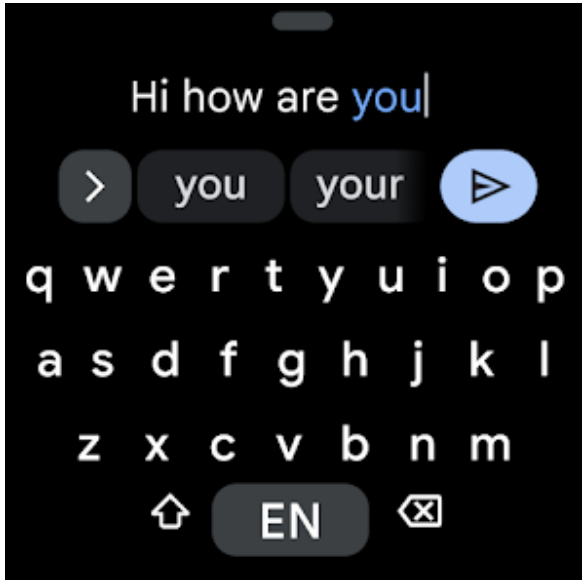
Gboard: Mobile Keyboard

Security Lancaster | Lancaster University

- **Language modeling : Federated RNN**
  - Predict the next word based on typed text
  - Power the predictions strip
- **Emoji prediction, Action prediction and Discovering new words**

- Recordings of user actions
- whether they click recommended word or not

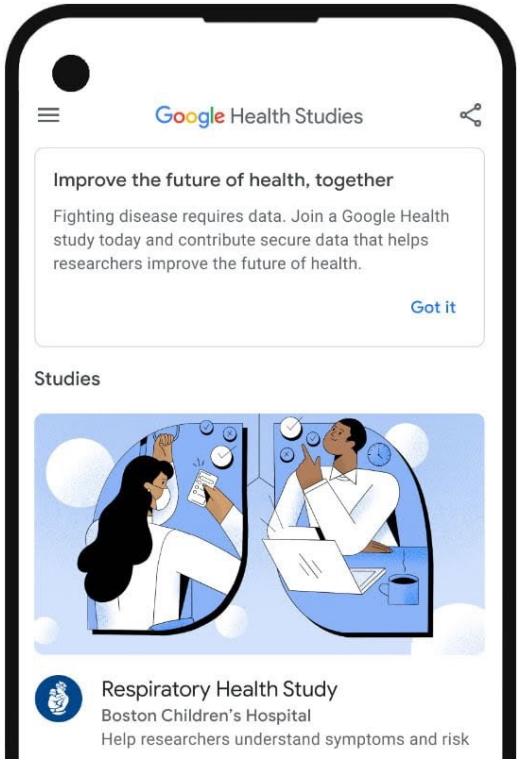




Google Health Studies

Security Lancaster | Lancaster University

- **Participate in research with Google Health Studies**



- User responses train the model which is then accumulated to create working average
- Responses can be withdrawn at anytime

FL on Google Pixel

Security Lancaster | Lancaster University

- **Now Playing: Song Recognition Measurement**
  - Computes the recognition rate
  - Secure aggregation protocol

- **User Settings**
  - Personalized preferences

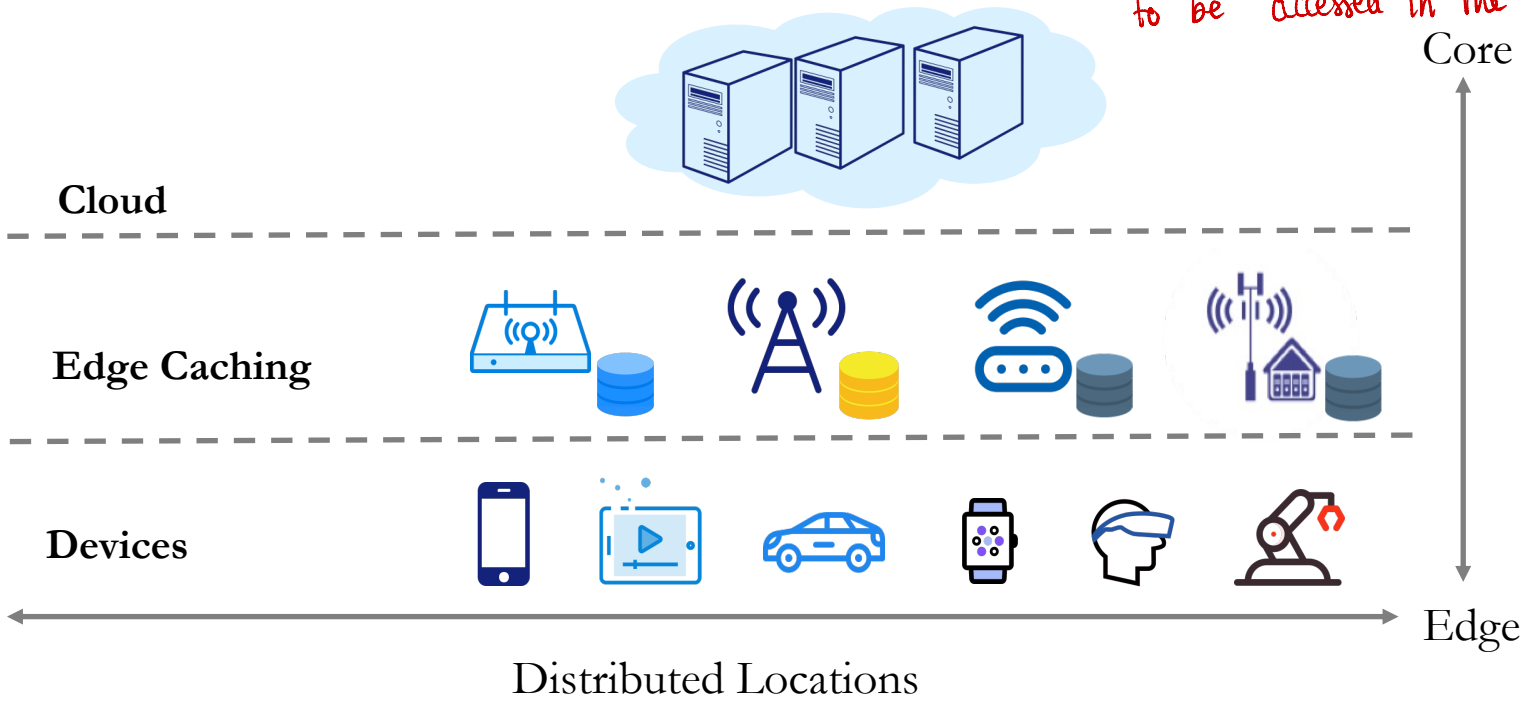
- Each device in a region will recieve the same song database
  - only these songs can be recognised
  - what songs are often recognised?
- When a song is recognised, the song name is recorded in the 'now playing' history, and it could be added to a playlist
- Client is then invited to join a federated communication route, where clients will compute a recognition rate, and secure aggregated protocols to encrypt the result, which is exploaded to server

Edge Caching

Security Lancaster | Lancaster University

- **Personalized Model**
  - More accurate content popular prediction
  - Proactive caching

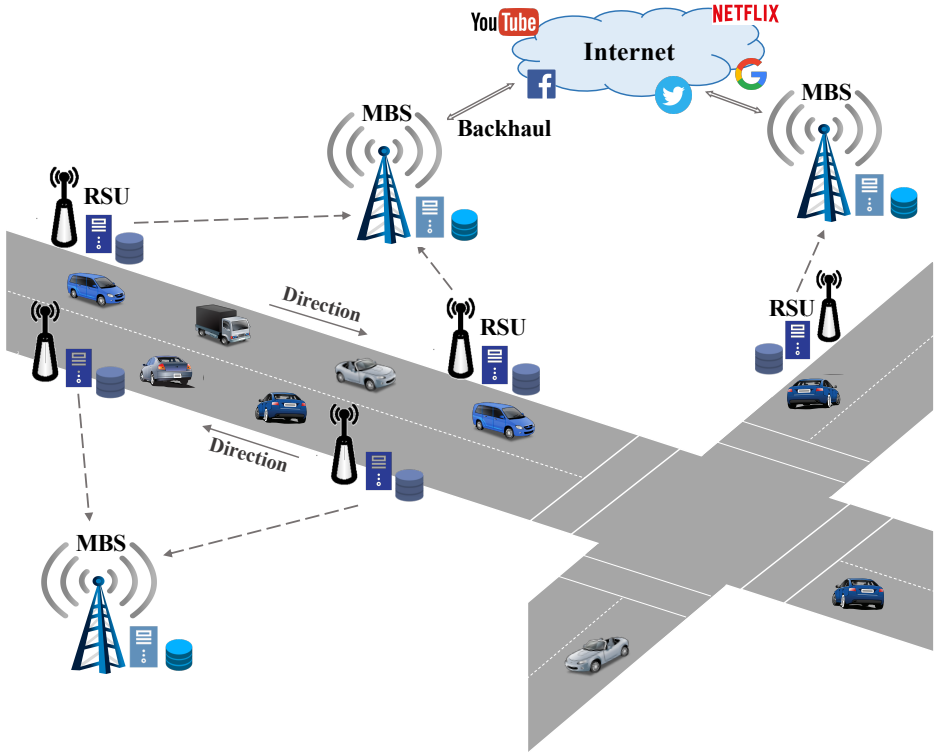
- Used to train models that predict which content is likely to be accessed at a particular area/time
  - Used to inform caching decisions on certain edge devices, allowing it to cache content that's more likely to be accessed in the future



Autonomous Vehicles

- **Real-time prediction:**
  - A better and safer self-driving car experience
  - E.g., Wheel steering angle prediction

data from other cars



Part 3: Cutting Edge Research in Secure FL

- An Adaptive and Robust Federated Learning Framework

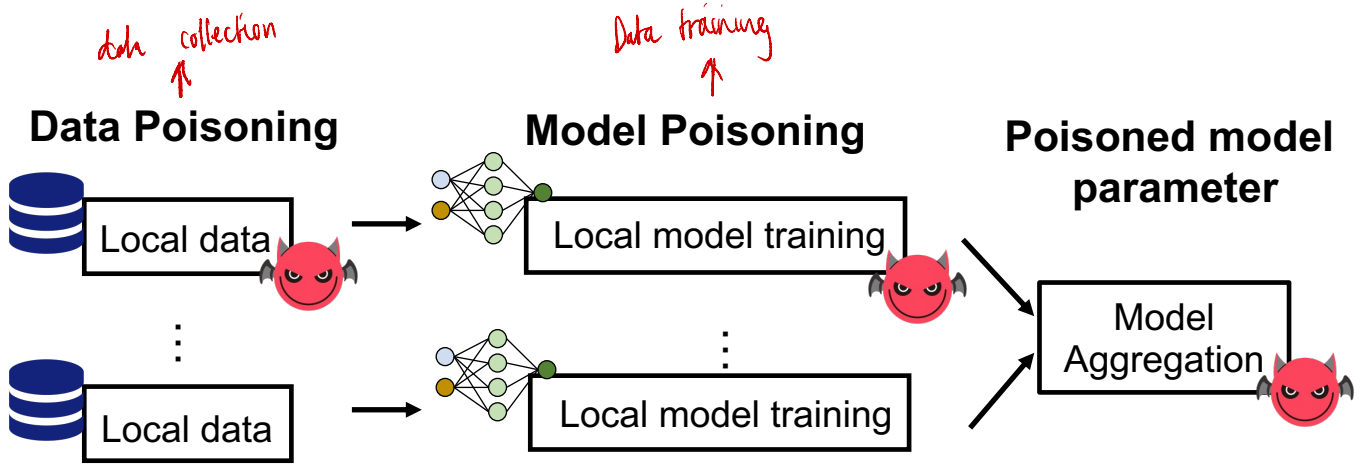
advantage: decoupling of training data from the need ...

Security for FL

Security Lancaster | Lancaster University

Adversarial attacks on FL

- Poisoning attacks → Attempts to modify models behavior in some undesirable manner
- Inference attacks → Targets user privacy



Goal of an attack:

(The target area of undesirable modification)

- Untargeted attacks → Aims to reduce the global model accuracy
- Targeted attacks → Aims to alter the model behavior on some samples while maintaining good behavior of other samples

Malicious Failures	Non-Malicious Failures
• Malicious Clients	• Noisy training labor • Users unknowingly uploading unreliable models due to high mobility and constraints

→ Design a robust FL framework

Challenges in Robust FL

Security Lancaster | Lancaster University

- Clients upload unreliable model updates intentionally or unintentionally
  - cant upsize useless data or control useless behavior
- Local resource heterogeneity
  - harder to process
- Hard to detect imperceptible attacks
  - very little inference in these attacks
- Large communication and computation cost
  Several attacks happening over time

SOTA Approaches

Security Lancaster | Lancaster University

- **Robust learning** – aggregate global model accuracy & tolerate small numbers of malicous clients
  - Aggregation rules

- **Malicious detection** – identify & remove malicous clients before the model aggregation occurs
  - Detect adversarial clients

- **Limitations**
  - Knowledge of client data distributions, IID training data
  - Server uses clean public dataset
  - Tolerate small number of malicious clients
  - Static prediction ruleset

dynamic

Require approach to detect malicious clients without prior knowledge

Objective

Security Lancaster | Lancaster University

- **RFML: A Robust-by-design FL Meta framework against adversarial attacks**
  - Reduce negative impact from malicious clients on non-IID data
  - Tuning DL architectures to defence attacks
  - Low communication/compute overhead

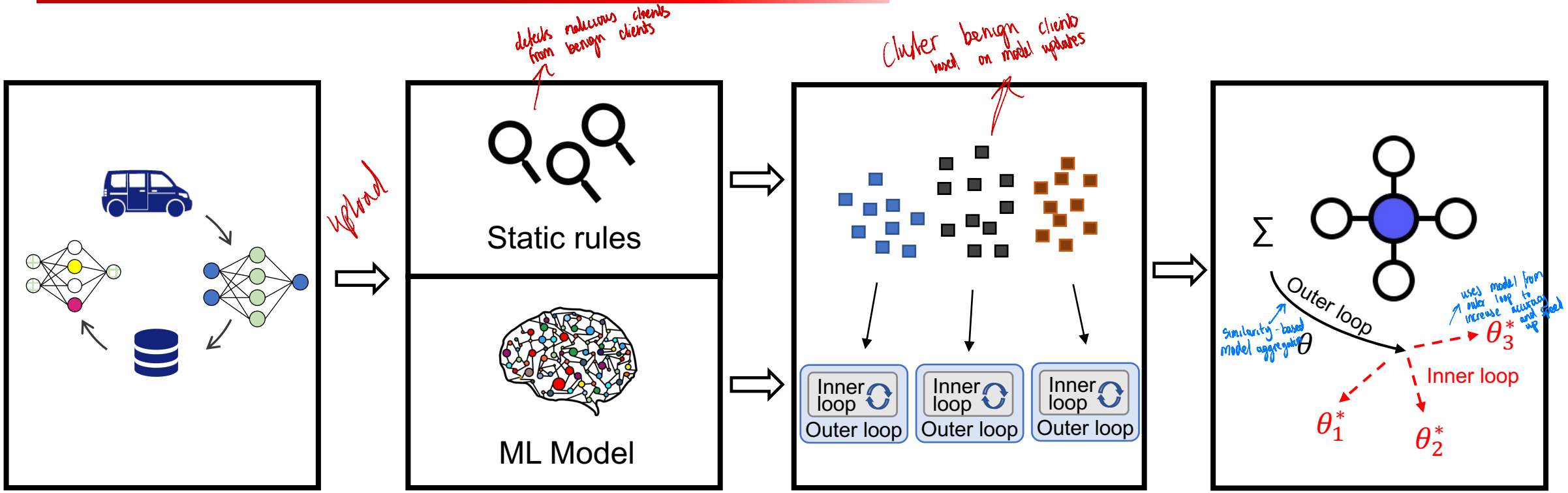
helps train the learning-based model

detects malicious clients

rule-based detection & learning-based model

- **Contributions:**
  - Federated-meta learning framework
  - Knowledge of malicious clients detection via ranking
  - Variational autoencoder (VAE) anomaly detection model
  - Similarity-based adaptive model aggregation method → reduce negative impact of malicious client

Robust-by-design FL Meta Framework



Local model training **Detection** **Cluster** **Model aggregation**

Malicious Clients Detection via Ranking

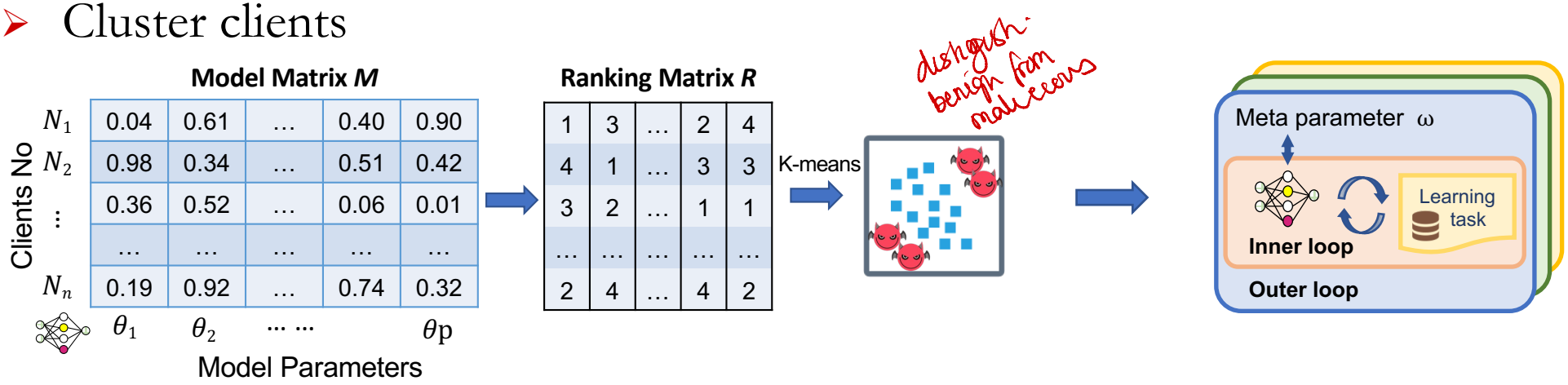
Security Lancaster | Lancaster University

Key characteristic of rank domain

- Transfer the original model updates into a ranking matrix
- Rank all clients along different parameter dimensions

Canonical correlation analysis (CCA) based K-means

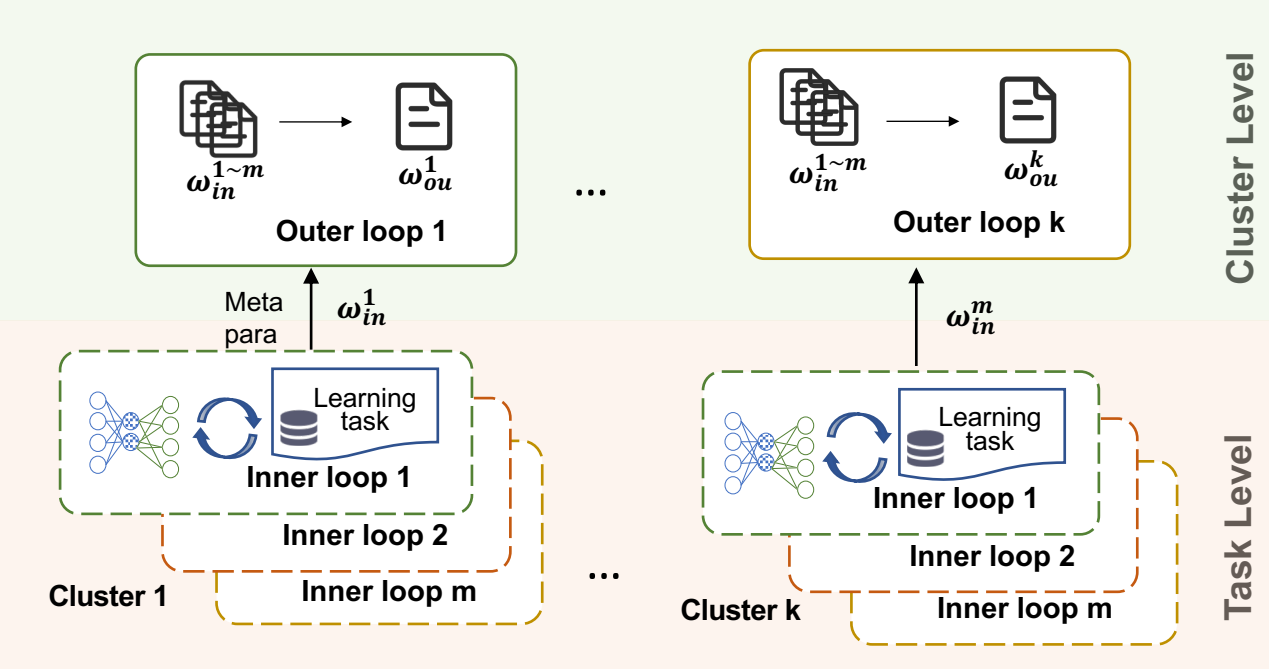
- Cluster clients



W. Zhu, B. Z. H. Zhao, S. Luo, and K. Deng, "Mandera: Malicious node detection in federated learning via ranking," arXiv preprint arXiv:2110.11736, 2021

Meta Learning based Approach to Data Characteristics

- **Learning to learn**
- **Train multiple models via learning loops** → from experience
  - Inner loop: Train task-specific model from local data
  - Outer loop: Extract common inner loop features, optimize meta-model adaptability
    - from similar tasks in each cluster



Learning based Detection

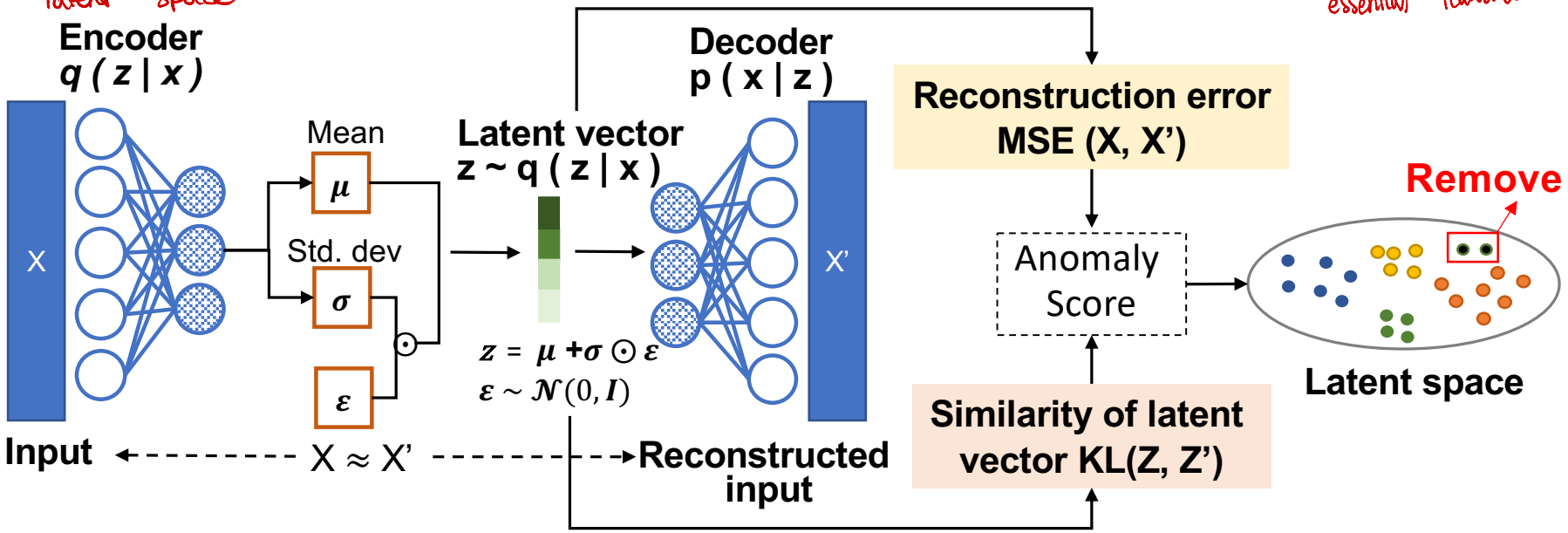
Security Lancaster | Lancaster University

- **Variational Auto-Encoder based detection model**
  - Low-dimensional embeddings
  - Reconstruction error + Similarity of latent vector

- Used to detect abnormal model updates based on low-dimensional embedding

- makes malicious clients harder to reconstruct from representation in the latent space

- easy to dete malicious from benign class since the essential features are too different



Similarity based Model Aggregation

Security Lancaster | Lancaster University

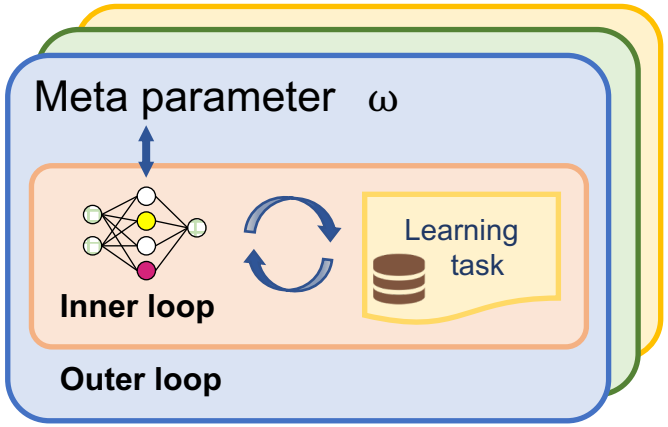
Calculate essential means & compute similarity score from new value or calculated meta models

Similarity Calculation

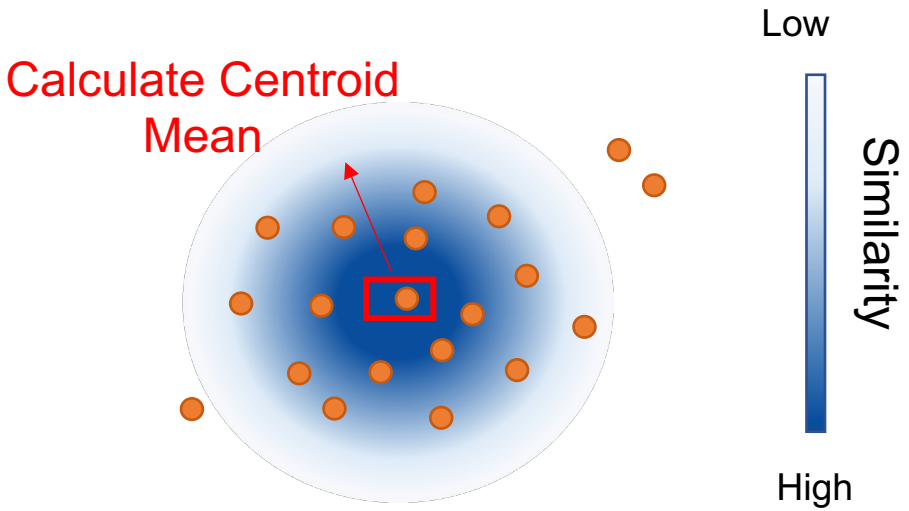
- Canonical Correlation Analysis (CCA) similarity
  - Measures similarity distance between model updates

Similarity-based adaptive meta aggregation

- Model-agnostic



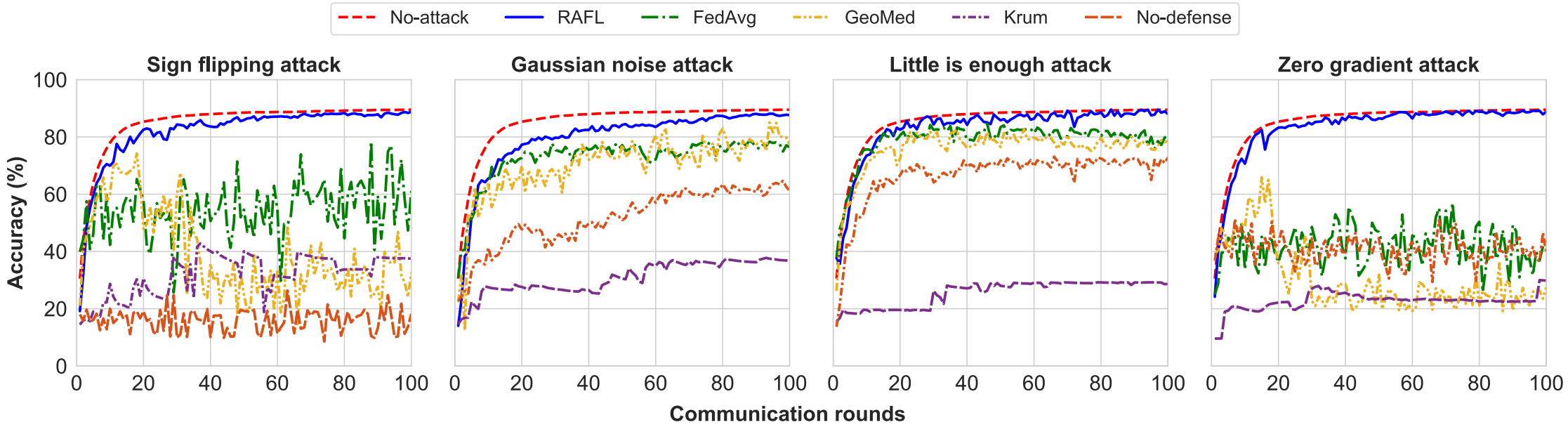
Clusters



Each Cluster

Experimental results

- The RAFL framework outperforms other baseline defensive methods against adversaries in terms of model accuracy and efficiency



Open Problems

- Attacks on FL and corresponding defenses
- Model heterogenous
- Optimization algorithms for FL (communication-efficient)
- Asynchronous model update in FL
- etc

Security Lancaster | Lancaster University

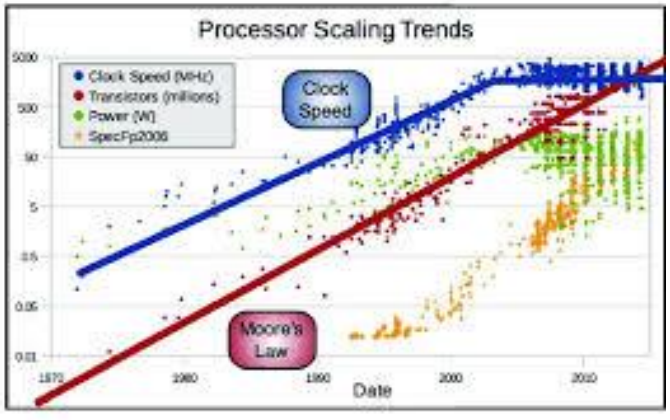
Thank you very much !

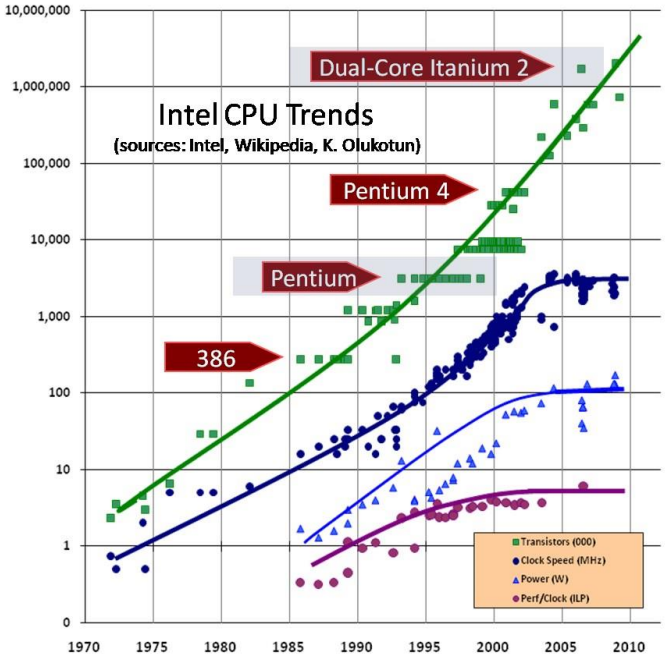
Security Quantification

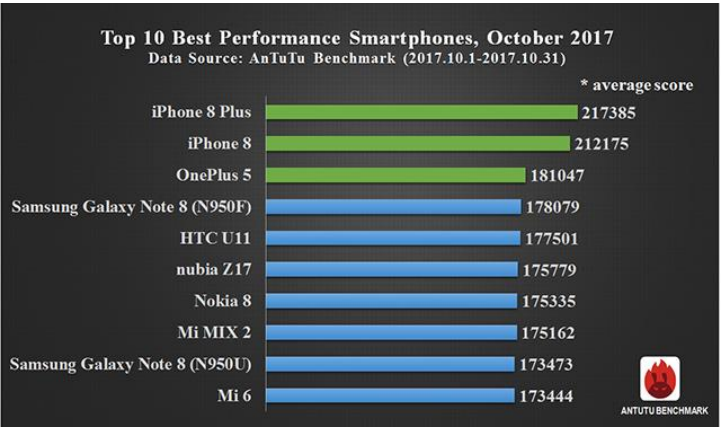
Take Away's from Lecture

- Notion of Security Metrics
- Notion of Attack Surfaces

Numbers, Numbers & Numbers







Security?

NEW!!!
93.5% Security!

Metrics and measurements needed

- To establish a baseline
- To establish as basis for comparing and improving

How secure is our software?

Which is more secure: Vista or Win10 or MacOS? Ubuntu or Fedora?

Are we better-off than before?

Over development (or for testing), how do we determine how much to invest for achieving the requisite level of security?
What is the bang for the buck - RoI?

Metrics

different processors w/ different capabilities to certain tasks

- **Performance**
  - Computing: MIPS: **M**illions of **I**nstructions **P**er **S**ec w.r.t chosen computational benchmarks e.g., matrix mult, R/W, image proc
  - DB's: # Transactions per sec w.r.t chosen benchmarks

- **Systems**
  - Availability: Duration of up-time w.r.t a given a set of failures
  - Reliability: Likelihood of delivering service w.r.t failure types

- **Security?** Next Slide

Security?

- **C.I.A:** Metrics or Attributes?
  - **C**onfidentiality (~Authorization Based Access | Binary)
  - **I**ntegrity (~Correctness & Consistency of Data | Binary)
  - **A**vailability (~Access or Degradation| Probability)

Meaningful only w.r.t to a specific (known) vulnerability and the degree of severity of the vulnerability

Options

- Effort Metrics (Absolute or Comparative)
  - Breaking crypto algorithms: SHA-128 etc – number of operations (~time/cost) required to break algorithm; breaking a password etc
  - Time/cost to probe for vulnerabilities & access – full or partial
  - …

Let's revisit the term "Security Quantification"

The degree of quantifiable trust one can expect for the delivery of the expected service(s) despite the encountered perturbations

Focus of Lecture:

1. Vulnerability Scoring: Assessing potential for damage (e.g. CVSS)
2. Attack Surfaces: Estimating the potential exposure to attacks

CIA: **C**, **I** are binary values; **A** is a probability so…

- How do we combine them?
- Should we expect a single number for security or a tuple?
- What does it mean to have X% security? ☺
- Similar to "w.r.t benchmarks", **C**, **I** and **A** are only meaningful w.r.t to a given vulnerability and its degree of severity, right?

Damage - Subjective Attribute

→ Depends on the value of assets & its usage (Priority)

→ Can be 'latent' - Harder to detect until until its too late to prevent any damage

→ Persistance of the vulnerability → May be difficult to categorise the threat

CVSS: Common Vulnerability Scoring System

- US NIST https://nvd.nist.gov/vuln-metrics/cvss
- Required for reporting a vulnerability conformal to the *CVE* (*Common Vulnerability Exposure*) grading reference

- CVSS → **A single number**!!!
  Composed from 3 metric groups

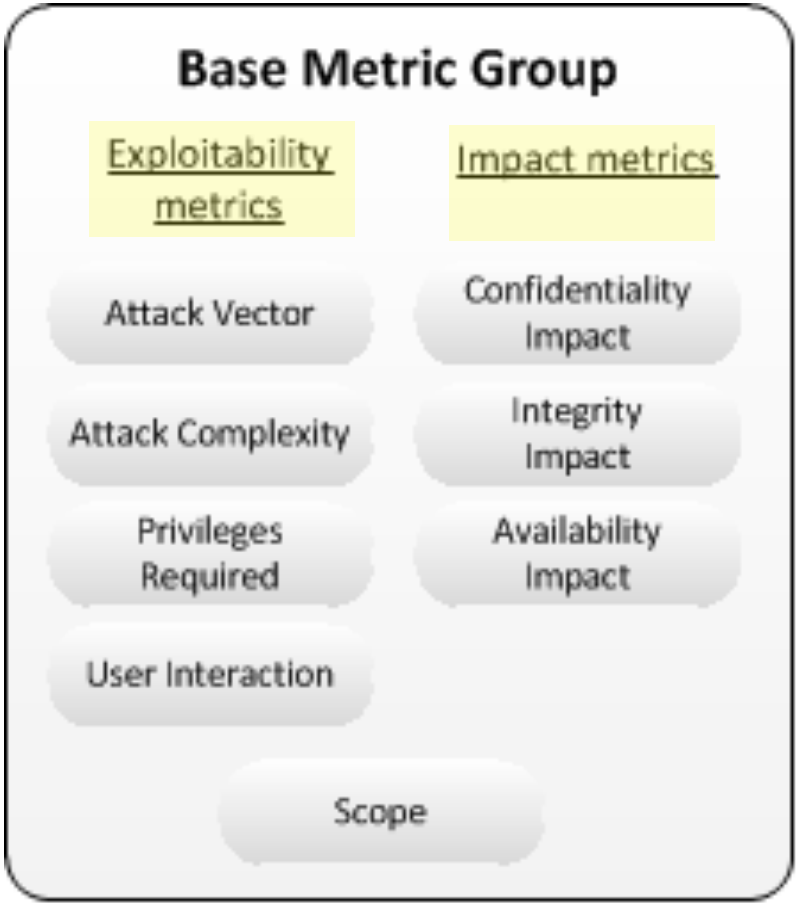
1. **Base Metrics**: No changes over time or environment
2. **Environmental Metrics**: Might vary over different deployments
3. **Temporal Metrics**: Might change over time

Aggregated to yield a single number as the weighted "average" (average, mean, median…) of the Base, Env. & Temporal Metrics

I. Base Metric Group

1. Exploitability metrics
   - Characterize the vulnerability
2. Impact metrics
   - Represent the consequences

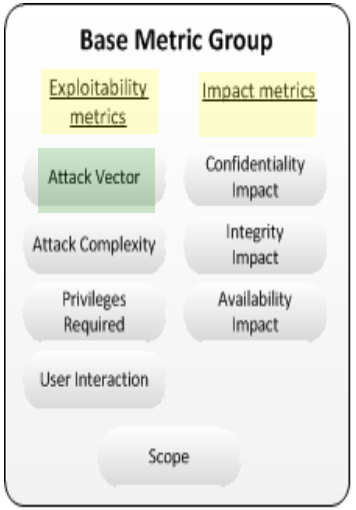
Exploit = f(AV, AC, PR, UI)
Impact = f(C, I, A)



Base: Attack Vector (AV)

- What are the means of exploitation?

  Metric Value:
  - (P) Physical
  - (L) Local only
  - (A) Adjacent network (e.g. Wifi, local IP subnet)
  - (N) Network: Remotely exploitable



Class

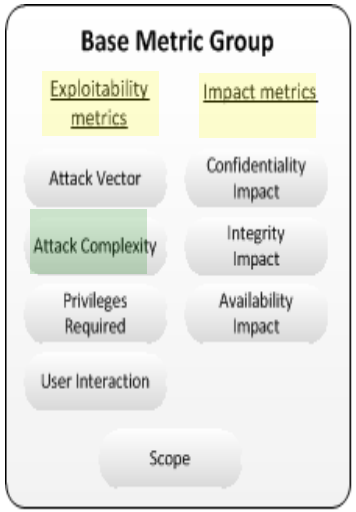
- More than one level affected? Go with the worst one
- Client that opens material from an untrusted internet source? e.g. Zip utility with a buffer overflow (N)
- Cross-site scripting-XSS injecting scripts in Webapp? (N)
- Lack of SSL encryption on Facebook client? (A/L)

→ Higher AV for more/wider spread of (logical, physical) attackers

Base: Attack Complexity (AC)

Subjective (Letter/Number Grade)

What is the amount of effort needed to conduct an attack



- How complex is the exploit?
  - One step? e.g. Buffer overflow
  - Multi-step? e.g. Phishing: Get user to click/download

- Metric value

  (H) High: Specialized access conditions
  - e.g. Overcoming advanced exploit mitigation techniques

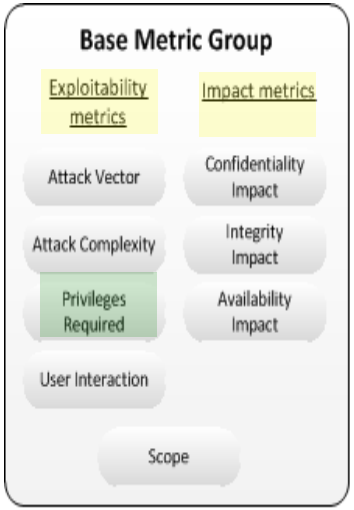
  (L) Low: No specialized conditions → Bad
  - e.g. Default configurations
  - e.g. Requires little skill to perform

→ <u>Low complexity is bad</u> as it implies least effort for an attack

Base: Privileges Required (PR)

- Level of privileges needed for exploit?

- Metric value
  (H) Privileges that provide significant control
  (e.g. Administrative)
  (L) Privileges that provide basic user capabilities
  (N) No authorization needed

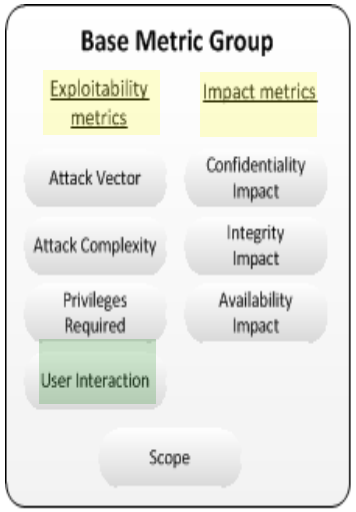


- In an authentication system itself?
  e.g. Kerberos style network authentication? (N)
  e.g. Path traversal in photo upload for a Twitter client? (L)
  e.g. Insecure pseudo random # gen for session IDs? (N)

→ PR damage is greatest if least privileges are required to get in

Base: User Interaction (UI)

- A user, other than the attacker, participates in the exploit

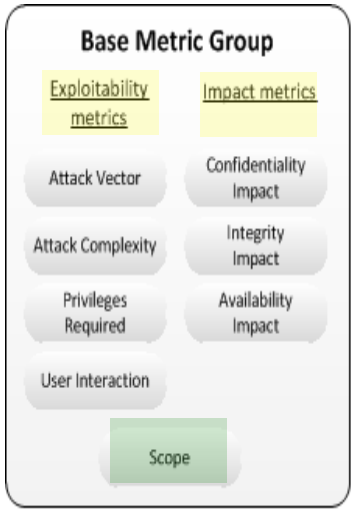


- Metric value
  (N): Can be exploited without interaction from user
  (R): Requires a user to take action
    e.g. exploit may only be possible during the installation of app by a system administrator.
- Automated scripting attack; buffer overflows (N)
- Must click on a link (R)
- Need a victim to forge the http request & be logged in (R)

→ UI has highest value when no user interaction is required

Base: Scope (S)

- The ability for a vulnerability in one software component to impact resources beyond its means, or privileges



- Metric Value
  (U): Unchanged
    - Vulnerable + impacted component are the same
  (C): Changed
    - Vulnerable + impacted component differ

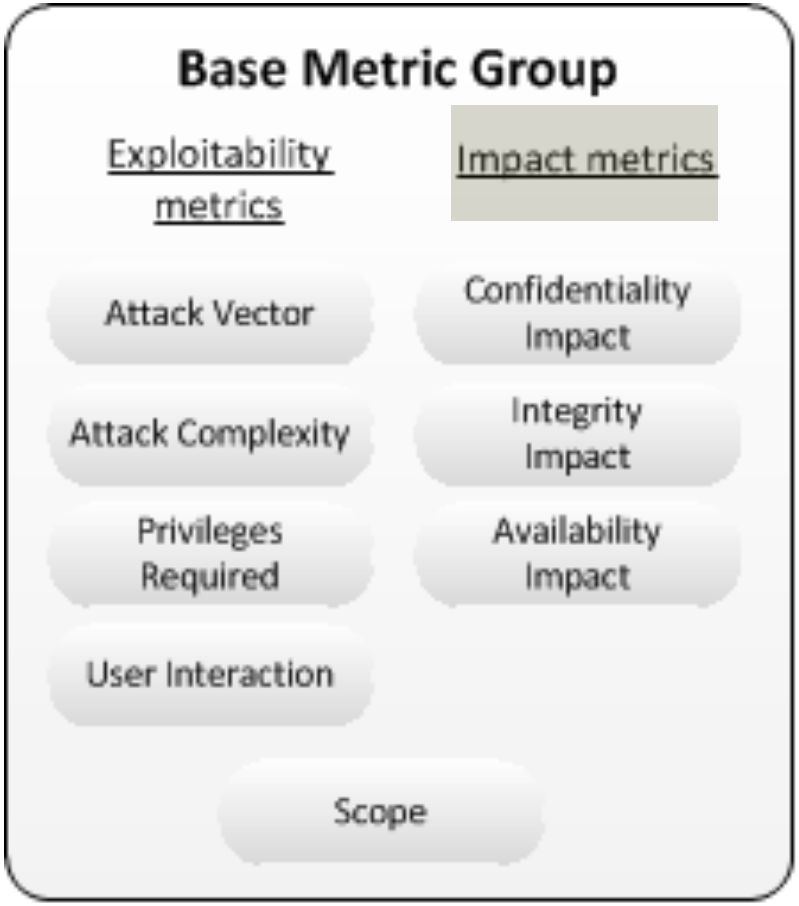
- Vulnerability in VM compromises host OS? Scope changed
- Hypervisor/TCB violations? Scope changed

→ Base score is greatest when a scope change has transpired

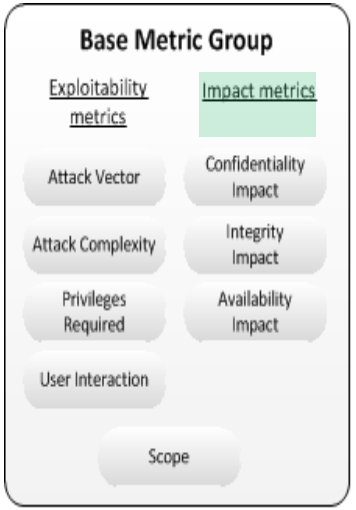
Base Metric Group

1. Exploitability metrics
   - Characterize the vulnerability
2. Impact metrics
   - Represent the consequences

- **Exploit** = f(AV, AC, PR, UI)
- **Impact** = f(C, I, A)



Base: CIA Impact



- Any impact on
  - Confidentiality, Integrity, and/or Availability?
  - ➢ These are three separate metrics!

- Metric Value (for each metric)
  (N) None
  (L) Low
    - e.g. Disclosing some database tables
    - e.g. Temporary DoS
  (H) High
    - e.g. Reading arbitrary memory locations is High Disclosure
    - e.g. Rule bypass of plug-in sandbox is High Integrity

E.g., Root-level access credentials compromised
→ Impact on C = High, I=High, A=High

Base Score Metrics

Exploitability Metrics

Attack Vector (AV)*

Network (AV:N)	Adjacent Network (AV:A)	Local (AV:L)	Physical (AV:P)

Attack Complexity (AC)*

Low (AC:L)	High (AC:H)

Privileges Required (PR)*

None (PR:N)	Low (PR:L)	High (PR:H)

User Interaction (UI)*

None (UI:N)	Required (UI:R)

Scope (S)*

Unchanged (S:U)	Changed (S:C)

Impact Metrics

Confidentiality Impact (C)*

None (C:N)	Low (C:L)	High (C:H)

Integrity Impact (I)*

None (I:N)	Low (I:L)	High (I:H)

Availability Impact (A)*

None (A:N)	Low (A:L)	High (A:H)

II. Environmental Metrics

physical Manifestation of the consequence of an attack eg. loss of life

1. **Base Metrics**: No changes over time or environment
2. **Environmental Metrics**: Might vary over different deployments
3. **Temporal Metrics**: Might change over time

- Security Requirements (CR, IR, AR) Confidentiality Requirement,...
  - Metric Value:

    (X): Not defined. Does not influence the score

    (H): A catastrophic adverse effect

    (M): A serious adverse effect

    (L): A limited adverse effect

- Often Base Metrics are <u>modified</u> with a multiplication factor
  - Loss of life, physical assets, productivity
  - Levels: None, Low, Medium, High, Not defined

III. Temporal: Exploitability (E)

1. **Base Metrics**: No changes over time or environment
2. **Environmental Metrics**: Might vary over different deployments
3. **Temporal Metrics**: Might change over time - Exploitability, Remediation, Confidence

- Is there a known public exploit?
- Metric Value

  (U) Unproven, entirely speculative exploit: SCA/CCA

  (POC) Proof-of-concept exists, but no known malicious exploits

  (F) Functional exploit is available

  (H) Functional exploit is widely disseminated

  (X) Not defined e.g., zero-day attacks

  - Being temporal, these could change quickly
  - White Hats (ethical hacker) could write exploits (pen. testing) to deliberately increase score (F), so that it actually gets fixed ☺

    Force the developers to pay attention / fix exploit

→ <u>The easier a vulnerability can be exploited, higher the vul. score</u>

Temporal: Remediation (RL) & Confidence (C)

- What is the *level of available remediation (RL)*?
  - (O) Official Fix is available
  - (TF) Temporary fix is available
  - (W) Workaround is available
    - Unofficial, non-vendor patches,
    - Temporary change in configuration
  - (U) Nothing is released yet
  - (X) Not defined

- What is the *report confidence (RC)*?
  - (U) Unconfirmed or there are multiple conflicting reports
  - (R) Significant details are published e.g. proof-of-concept
  - (C) Confirmed by the source
  - (ND) Not defined

CVSS Scoring Tips

- Metric groups
  1. **Base Metrics**: No changes over time or environment
  2. **Environmental Metrics**: Might vary over different deployments
  3. **Temporal Metrics**: Might change over time

- Score individually – ignore interactions with other vulnerabilities
- Assume the most common or default configuration of the server
- Score the greatest exploitation impact, if there are many
- Emphasise targets to the host, not really other users

➔ Take <u>weighted</u> aggregate (aggregate: average, mean, median…)

e.g. CVE-2015-7197

- Firefox improperly controls the ability of a Web worker to create a WebSocket object, which allows remote attackers to bypass intended Mixed-content restrictions via crafted JavaScript code

Web worker long-running Javascript tasks that do not interrupt the user

Mixed-content restrictions: Cannot use one WebSocket for multiple protocols (e.g. both HTTPS and SSH)

CVSS of CVE-2015-7197

- AV: Access Vector? Network
- AC: Access Complexity? Low
- PR: Privileges req? None
- UI: User interaction? None (in general)

- C: Confidentiality? None. Possibly low depending on access
- I: Integrity? Low. Allows access to sockets
- A: Availability? None

Base Score Metrics

Exploitability Metrics

Attack Vector (AV)*

Network (AV:N)	Adjacent Network (AV:A)	Local (AV:L)	Physical (AV:P)

Attack Complexity (AC)*

Low (AC:L)	High (AC:H)

Privileges Required (PR)*

None (PR:N)	Low (PR:L)	High (PR:H)

User Interaction (UI)*

None (UI:N)	Required (UI:R)

Scope (S)*

Unchanged (S:U)	Changed (S:C)

Impact Metrics

Confidentiality Impact (C)*

None (C:N)	Low (C:L)	High (C:H)

Integrity Impact (I)*

None (I:N)	Low (I:L)	High (I:H)

Availability Impact (A)*

None (A:N)	Low (A:L)	High (A:H)

So how to grade them? Weighting

- Weights derived as:
  - Experts get together
  - Analyze a bunch of vulnerabilities in their products
  - Agree on the labels for each vulnerability
  - Agree on an overall ranking of many previous vulnerabilities
  - Adjust the weights to match

- CVSS writers recommend
  - NIST calculator https://www.first.org/cvss/calculator/3.0
  - Report each level along with the weighting
    - Weights may evolve
    - Comparing two vulnerabilities can also be grades vs. numbers

Base Weights (Table or Weighted Aggregate)

Metric	Metric Value	Numerical Value
Attack Vector / Modified Attack Vector	Network	0.85
	Adjacent Network	0.62
	Local	0.55
	Physical	0.2
Attack Complexity / Modified Attack Complexity	Low	0.77
	High	0.44
Privilege Required / Modified Privilege Required	None	0.85
	Low	0.62 (0.68 if Scope / Modified Scope is Changed)
	High	0.27 (0.50 if Scope / Modified Scope is Changed)
User Interaction / Modified User Interaction	None	0.85
	Required	0.62
C,I,A Impact / Modified C,I,A Impact	High	0.56
	Low	0.22
	None	0

Alternative: CWSS

- Common ***WEAKNESS*** Scoring System
  - More detailed, but not as widely-adopted
  - http://cwe.mitre.org/cwss/
  - Categories: Base, Attack Surface, Environmental

- Base Finding Metric Group
  - e.g. Acquired Privilege, Acquired Privilege Layer
    - *Access to Network, App, entire Enterprise*
  - e.g. Internal Control Effectiveness
    - *Would detection (of exploit) measures have been effective?*

Not essential (Jargon)

- Attack Surface Metric Group
  - e.g. Required Privilege AND Required Privilege Layer
    - *How much authentication was needed?*
  - e.g. Level of Interaction
    How much social engineering is required?
- Environmental Impact Group
  e.g. Business Impact, Likelihood of Discovery/Exploit, Remediation Effort

Attack Surface Measurement (ASM)

How do we quantify an artifact's security?

Measure its exposure to attacks → # of attack entry points, type/severity of attacks : Attack Surface

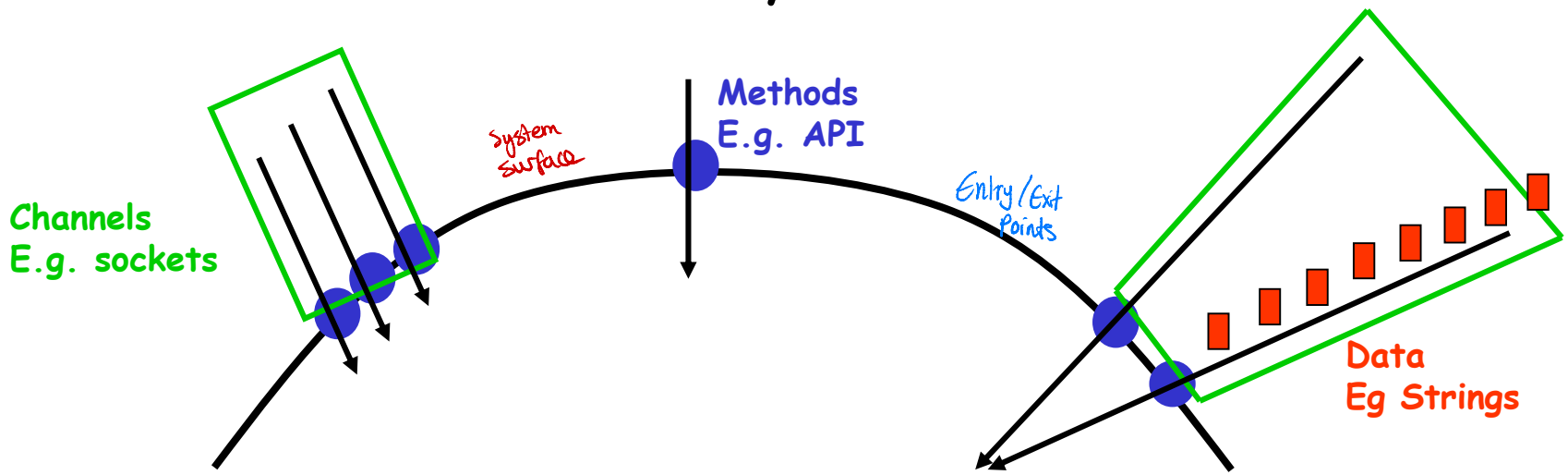
How hard are they to exploit? What's the damage range?

- HW: # of open ports, covert or side-channels,…
- Network: # of malicious packets,…
- Software level?
- System level?

Attack Surfaces

exit points as well

Attacks Entry Points



The attack surface of a system ='s all the ways in which an adversary can enter the system and potentially cause damage

<u>Attack Surface Measurement</u>: Identify relevant resources (methods, channels, and data) + estimate the contribution of each such resource

The Needs

- Formalize the notion of attack surfaces
- Introduce a systematic attack surface measurement method
- Validate the method → Essential

❑ Not All Resources Part of the Attack Surface
  - Only those resources that the attacker can use to send data into or receive data from the system are relevant
  - Framework for entry point and exit points to identify resource access
  - Depend if they play a role in the input/output

❑ Identify Relevant Resources
  - Entry Points and Exit Points
    - Static analysis
    - C library methods (e.g., read) for data exchange
    - Call graphs

Tools

  - Channels and Untrusted Data Items
    - Identification of data channels – data propagation paths
    - Run time monitoring on data-types/access-modes
    - Volume/types of Data reads and Writes

No claims on completeness
KK, KU, UU's!

Attack Surface Measurement (ASM)

1) Framework to identify a set M of entry Methods; a set C of Channels; a set DI of untrusted Data Items ➔ AS ~ <M, C, DI>

2) Estimate a resource's contribution to the attack surface as a ratio of Damage-Potential vs. Effort-needed-for-Attack

Resource	Damage Potential	Effort for Attack
Method	Enhanced Privilege	Access Rights
Channel	Protocol Violation	Access Rights
Data Items	Date Type Errors	Access Rights

Reduce attack surface to (a) increase the difficulty and (b) decrease impact of exploits (assuming linear damage correlation)

→ Damage proportional to the size of the AS

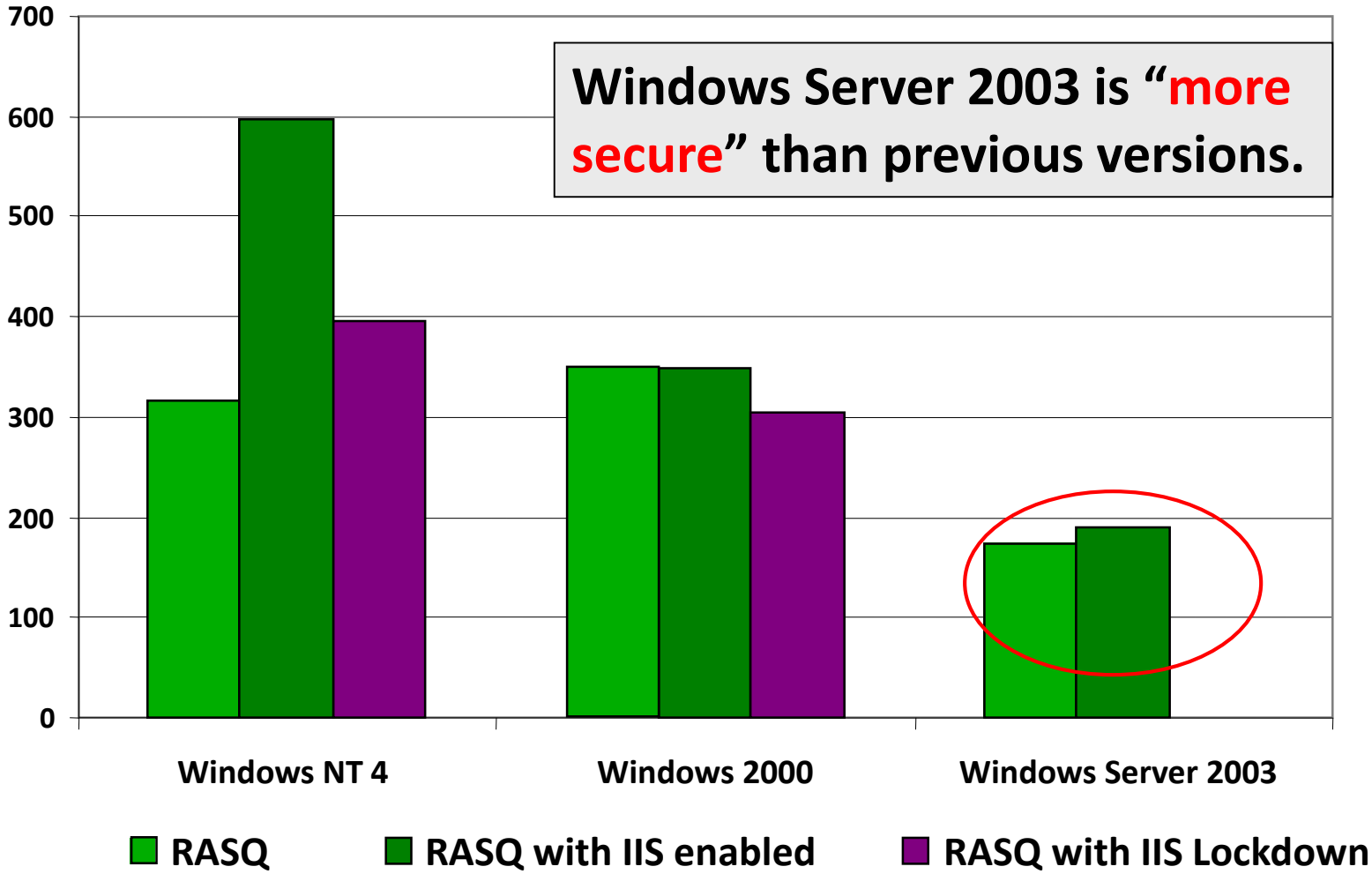
Linux Attack Surface Measurements

Attack Vector	Debian	RH Default	RH Facilities	RH Used
Open socket	15	12	40	41
Open RPC endpoint	3	3	3	3
Services running as root	21	26	29	30
Services running as nonroot	3	6	8	8
Setuid root programs	54	54	72	72
Local user accounts	21	25	33	34
User id = root accounts	0	4	3	3
Unpassworded accounts	0	0	2	2
Nobody account	1	1	1	1
Weak file permission	7	7	21	37
Scripts enabled	1	2	2	2

Supports <u>perception</u> that Debian is more secure than RedHat

Ad hoc selection of parameters and measurements

Relative AS Quotient (RASQ)



Ad hoc selection of parameters and measurements

Validating the Prediction System

- If system A is more secure than system B, then ASM(A) < ASM(B)
- **Assumption**: Vulnerability patches improve software security

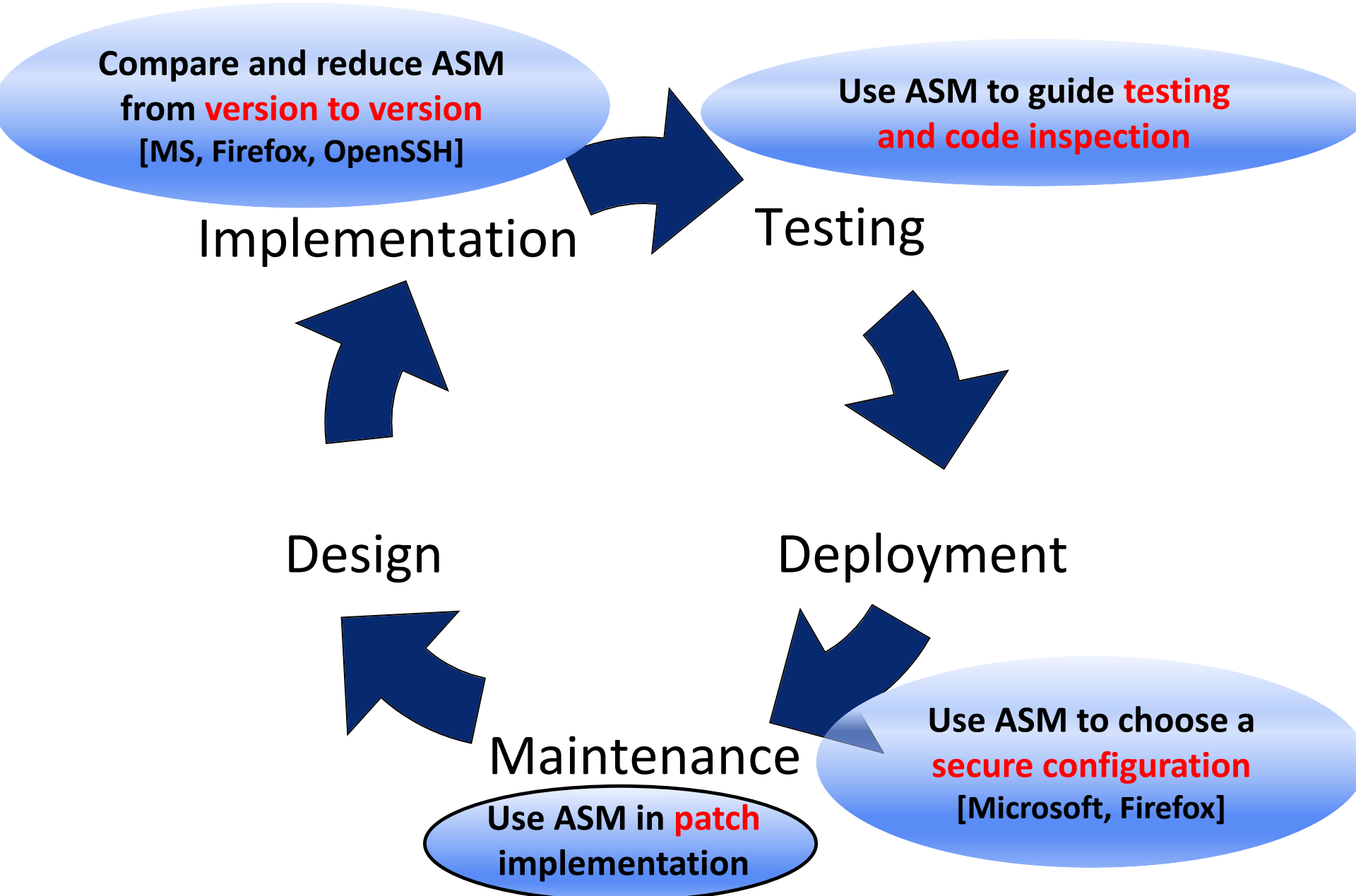
Thus, if ASM(After Patch) < ASM(Before Patch) ➔ ASM works

Assertion: Patches should reduce the attack surface

Software	% of Patches reducing ASM
Firefox 2.0	67%
ProFTP (all)	70%
All NVD Bulletins	77%

no definite

ASM in Software Development LifeCycle

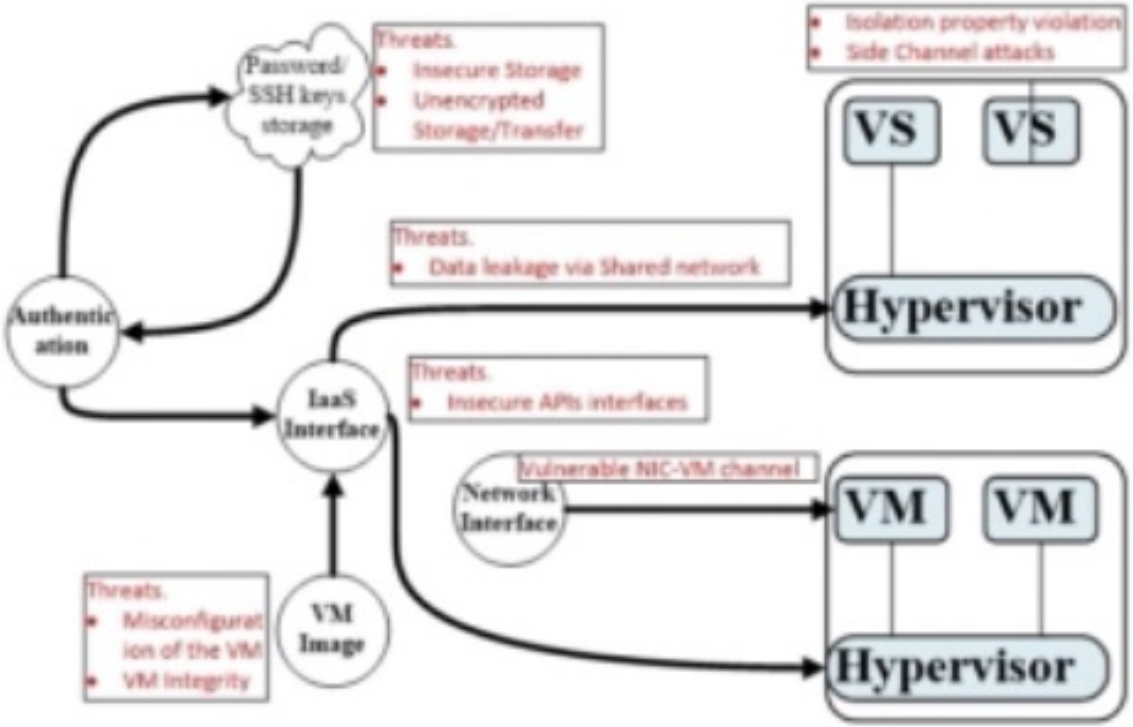


SW to Systems to Cloud: The Problem

- Multi-layer multi-component operations
- VM and Brokers
- Data resides on Cloud Service Providers (CSP) – hard to do ASM

- Identify functional components of the Cloud and their inter-relations
- Difficult to assess vulnerabilities, e.g CVSS for what function?
  - ✓ How does a User assess a CSP's ability to meet it's security requirements, and select a CSP on this basis?
  - ✓ What are the trades-offs across CSPs?
  - ✓ How does one monitor/enforce the agreed security levels with the CSP?

Functionality Modelling: Component by Component



- **Threats for Network Layer**
  - Shared network: eavesdropping
  - Vulnerable Hypervisor-VM, Network-VM communication
  - …
- **Virtualization Layer**
  - VM Isolation violation
  - VM Management exploits
  - VM authenticity, VM software Integrity, VM migration
  - …

Synopsis

- Sec Q is only relevant w.r.t (known) vulnerabilities
- Sec Q is hard: Typically synthetic range
  - No absolute #'s! Comparative
  - No completeness claims
  - Very limited coverage of collusion/cascade attacks
- Hard issue: How can a user do measurements to validate claims?

Developers would give specifications of attacks

Summary

- Basic notion of security metrics: only relevant w.r.t (known) vulnerabilities
- Quantification is hard: Typically, a synthetic range
  - No absolute #'s: Typically comparative
  - C.I.A are varied in dimension and type (binary, probabilistic)
  - No completeness claims

- Open Issues:
  - Cloud attack surfaces
  - How to validate security claims? By user?