SECURITY

Eunsuk Kang

Required reading: *Building Intelligent Systems: A Guide to Machine Learning Engineering*, G. Hulten (2018), Chapter 25: Adversaries and Abuse. *The Top 10 Risks of Machine Learning Security*, G. McGraw et al., IEEE Computer (2020).

LEARNING GOALS

- Explain key concerns in security (in general and with regard to ML models)
- Analyze a system with regard to attacker goals, attack surface, attacker capabilities
- Describe common attacks against ML models, including poisoning and evasion attacks
- Understand design opportunities to address security threats at the system level
- Identify security requirements with threat modeling
- Apply key design principles for secure system design

SECURITY: (VERY BRIEF) OVERVIEW

ELEMENTS OF SECURITY

ELEMENTS OF SECURITY

- Security requirements (also called policies)
    - What does it mean for my system to be secure?

ELEMENTS OF SECURITY

- Security requirements (also called policies)
    - What does it mean for my system to be secure?
- Threat model
    - What are the attacker's goals, capabilities, and incentives?

ELEMENTS OF SECURITY

- Security requirements (also called policies)
  - What does it mean for my system to be secure?
- Threat model
  - What are the attacker's goals, capabilities, and incentives?
- Attack surface
  - Which parts of the system are exposed to the attacker?

ELEMENTS OF SECURITY

- Security requirements (also called policies)
    - What does it mean for my system to be secure?
- Threat model
    - What are the attacker's goals, capabilities, and incentives?
- Attack surface
    - Which parts of the system are exposed to the attacker?
- Defense mechanisms (mitigiations)
    - How do we prevent the attacker from compromising a security requirement?

SECURITY REQUIREMENTS

- "CIA triad" of information security
- **Confidentiality**: Sensitive data must be accessed by authorized users only
- **Integrity**: Sensitive data must be modifiable by authorized users only
- **Availability**: Critical services must be available when needed by clients

EXAMPLE: COLLEGE ADMISSION SYSTEM

FEATURE

Hacker helps applicants breach security at top business schools

Among the institutions affected were Harvard, Duke and Stanford

Using the screen name "brookbond," the hacker broke into the online application and decision system of ApplyYourself Inc. and posted a procedure students could use to access information about their applications before acceptance notices went out.

CONFIDENTIALITY, INTEGRITY, OR AVAILABILITY?

CONFIDENTIALITY, INTEGRITY, OR AVAILABILITY?

- Applications to the program can only be viewed by staff and faculty in the department.

CONFIDENTIALITY, INTEGRITY, OR AVAILABILITY?

- Applications to the program can only be viewed by staff and faculty in the department.
- The application site should be able to handle requests on the day of the application deadline.

CONFIDENTIALITY, INTEGRITY, OR AVAILABILITY?

- Applications to the program can only be viewed by staff and faculty in the department.
- The application site should be able to handle requests on the day of the application deadline.
- Application decisions are recorded only by the faculty and staff.

CONFIDENTIALITY, INTEGRITY, OR AVAILABILITY?

- Applications to the program can only be viewed by staff and faculty in the department.
- The application site should be able to handle requests on the day of the application deadline.
- Application decisions are recorded only by the faculty and staff.
- The acceptance notices can only be sent out by the program director.

OTHER SECURITY REQUIREMENTS

- Authentication (no spoofing): Users are who they say they are
- Non-repudiation: Every change can be traced to who was responsible for it
- Authorization (no escalation of privilege): Only users with the right permissions can access a resource/perform an action

THREAT MODELING

WHY THREAT MODEL?

WHAT IS THREAT MODELING?

- Threat model: A profile of an attacker
    - **Goal**: What is the attacker trying to achieve?
    - **Capability**:
        - Knowledge: What does the attacker know?
        - Actions: What can the attacker do?
        - Resources: How much effort can it spend?
    - **Incentive**: Why does the attacker want to do this?

"If you know the enemy and know yourself, you need not fear the result of a hundred battles."
- Sun Tzu, The Art of War

ATTACKER GOAL

- What is the attacker trying to achieve?

ATTACKER GOAL

- What is the attacker trying to achieve?
    - Undermine one or more security requirements

ATTACKER GOAL

- What is the attacker trying to achieve?
    - Undermine one or more security requirements
- Example: College admission

ATTACKER GOAL

- What is the attacker trying to achieve?
  - Undermine one or more security requirements
- Example: College admission
  - Access other applicants info without being authorized

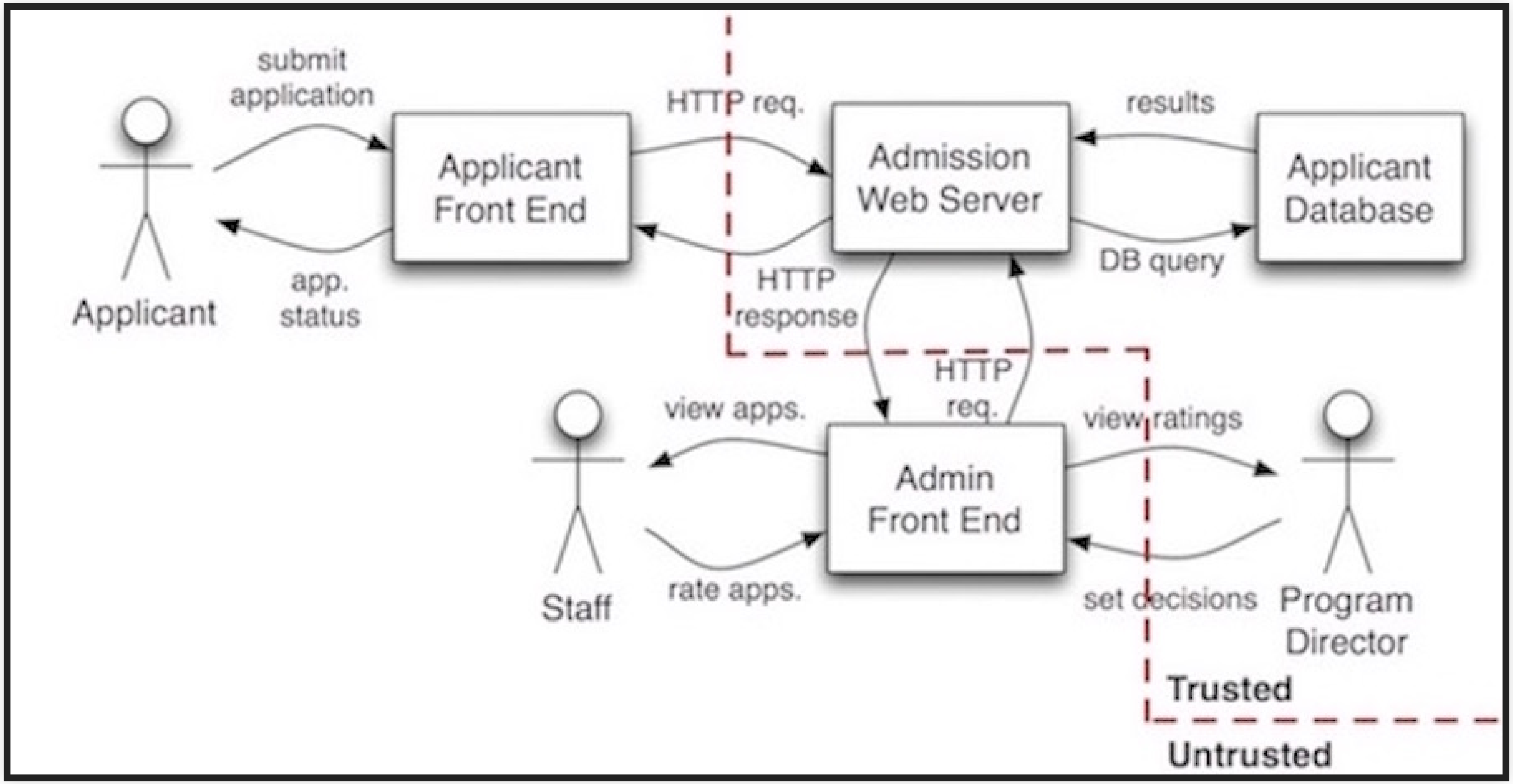
ATTACKER GOAL

- What is the attacker trying to achieve?
    - Undermine one or more security requirements
- Example: College admission
    - Access other applicants info without being authorized
    - Modify application status to “accepted”

ATTACKER GOAL

- What is the attacker trying to achieve?
    - Undermine one or more security requirements
- Example: College admission
    - Access other applicants info without being authorized
    - Modify application status to “accepted”
    - Cause website shutdown to sabotage other applicants

ATTACKER CAPABILITY



- What are the attacker's possible actions?
  - Depends on system boundary & its exposed interfaces
  - Use an architecture diagram to identify attack surface & actions

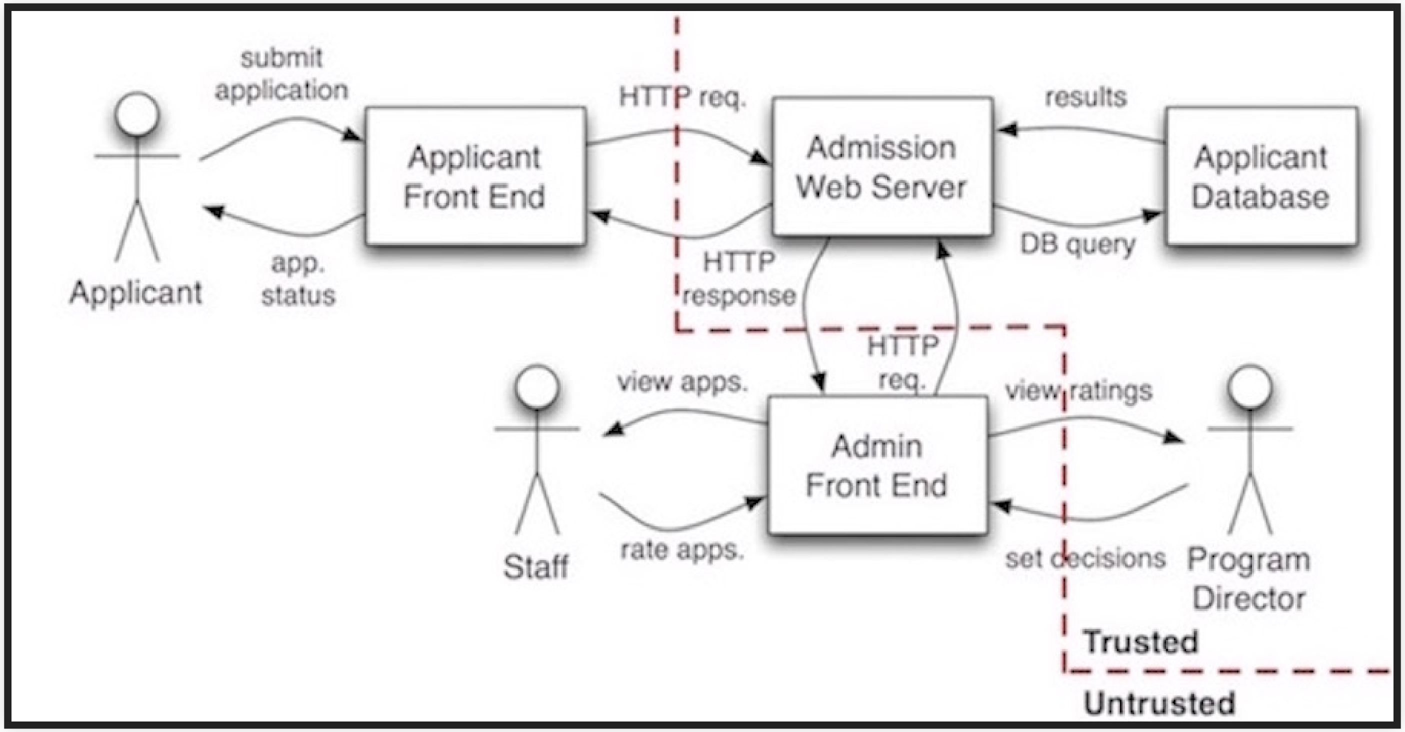
STRIDE THREAT MODELING

	Threat	Property Violated	Threat Definition
S	Spoofing identify	Authentication	Pretending to be something or someone other than yourself
T	Tampering with data	Integrity	Modifying something on disk, network, memory, or elsewhere
R	Repudiation	Non-repudiation	Claiming that you didn't do something or were not responsible; can be honest or false
I	Information disclosure	Confidentiality	Providing information to someone not authorized to access it
D	Denial of service	Availability	Exhausting resources needed to provide service
E	Elevation of privilege	Authorization	Allowing someone to do something they are not authorized to do

- A systematic approach to identifying threats (i.e., attacker actions)
  - Construct an architectural diagram with components & connections
  - For each component, enumerate & identify potential threats
  - For each potential threat, devise a mitigation strategy

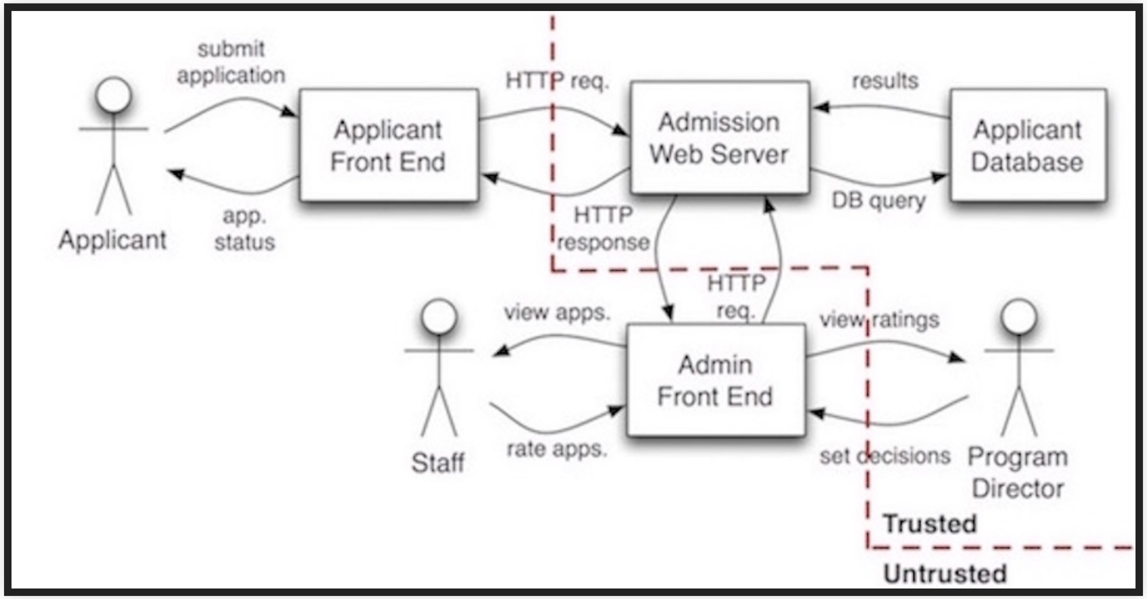
More info: STRIDE approach

STRIDE: COLLEGE ADMISSION



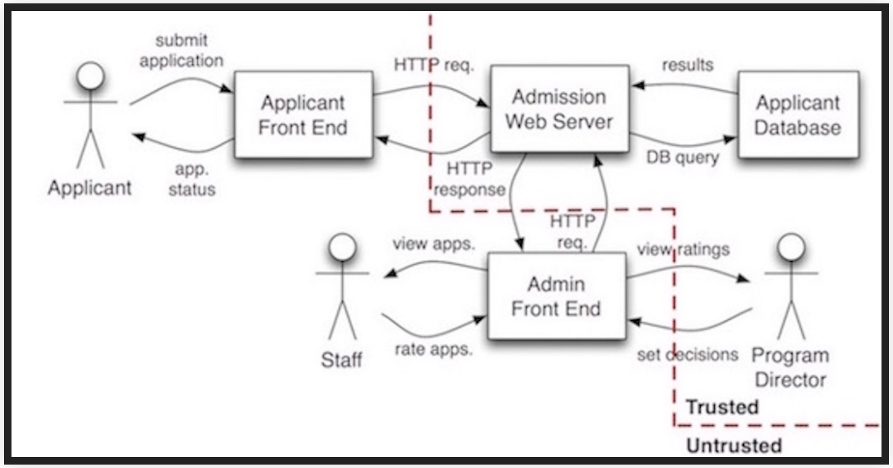
- Example: Application Front End
  - Spoofing: ?
  - Tampering: ?
  - Information disclosure: ?
  - Denial of service: ?

STRIDE: COLLEGE ADMISSION



- Example: Application Front End
  - Spoofing: Attacker pretends to be another applicant by logging in
  - Tampering: Attacker modifies applicant info using browser exploits
  - Information disclosure: Attacker intercepts HTTP requests from/to server to read applicant info
  - Denial of service: Attacker creates a large number of bogus accounts and overwhelms system with requests

STRIDE: MITIGATIONS



- Example: Application Front End
  - Spoofing: Attacker pretends to be another applicant by logging in -> **Require stronger passwords**
  - Tampering: Attacker modifies applicant info using browser exploits -> **Add server-side security tokens**
  - Information disclosure: Attacker intercepts HTTP requests from/to server to read applicant info -> **Use encryption (HTTPS)**
  - Denial of service: Attacker creates a large number of bogus accounts and overwhelms system with requests -> **Limit registration per IP address**

STRIDE & OTHER THREAT MODELING METHODS

	Threat	Property Violated	Threat Definition
S	Spoofing identify	Authentication	Pretending to be something or someone other than yourself
T	Tampering with data	Integrity	Modifying something on disk, network, memory, or elsewhere
R	Repudiation	Non-repudiation	Claiming that you didn't do something or were not responsible; can be honest or false
I	Information disclosure	Confidentiality	Providing information to someone not authorized to access it
D	Denial of service	Availability	Exhausting resources needed to provide service
E	Elevation of privilege	Authorization	Allowing someone to do something they are not authorized to do

- A systematic approach to identifying threats & attacker actions
- Limitations:
  - May end up with a long list of threats, not all of them critical
    - Think cost vs. benefit trade-offs: Implementing mitigations add to development cost and complexity
    - Focus on most critical threats
  - False sense of security: STRIDE does not imply completeness!

OPEN WEB APPLICATION SECURITY PROJECT

OWASP Top 10 Application Security Risks - 2017

A1:2017-Injection

Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

A2:2017-Broken Authentication

Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.

A3:2017-Sensitive Data Exposure

Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.

A4:2017-XML External Entities (XXE)

Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

A5:2017-Broken Access Control

Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

A6:2017-Security Misconfiguration

Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.

- OWASP: Community-driven source of knowledge & tools for web security

THREAT MODELING FOR ML

ML ATTACKER GOAL

ML ATTACKER GOAL

- Confidentiality attacks: Exposure of sensitive data

ML ATTACKER GOAL

- Confidentiality attacks: Exposure of sensitive data
    - Infer a sensitive label for a data point (e.g., hospital record)

ML ATTACKER GOAL

- Confidentiality attacks: Exposure of sensitive data
    - Infer a sensitive label for a data point (e.g., hospital record)
- Integrity attacks: Unauthorized modification of data

ML ATTACKER GOAL

- Confidentiality attacks: Exposure of sensitive data
  - Infer a sensitive label for a data point (e.g., hospital record)
- Integrity attacks: Unauthorized modification of data
  - Induce a model to misclassify data points from one class to another

ML ATTACKER GOAL

- Confidentiality attacks: Exposure of sensitive data
    - Infer a sensitive label for a data point (e.g., hospital record)
- Integrity attacks: Unauthorized modification of data
    - Induce a model to misclassify data points from one class to another
    - e.g., Spam filter: Classify a spam as a non-spam

ML ATTACKER GOAL

- Confidentiality attacks: Exposure of sensitive data
  - Infer a sensitive label for a data point (e.g., hospital record)
- Integrity attacks: Unauthorized modification of data
  - Induce a model to misclassify data points from one class to another
  - e.g., Spam filter: Classify a spam as a non-spam
- Availability attacks: Disruption to critical services

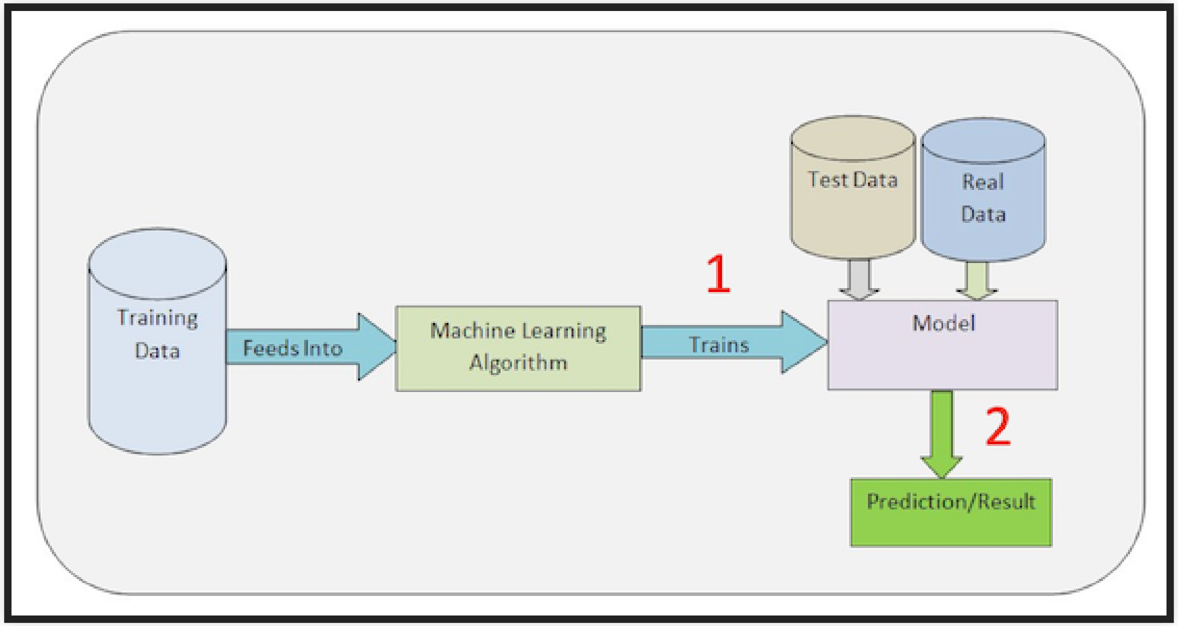
ML ATTACKER GOAL

- Confidentiality attacks: Exposure of sensitive data
  - Infer a sensitive label for a data point (e.g., hospital record)
- Integrity attacks: Unauthorized modification of data
  - Induce a model to misclassify data points from one class to another
  - e.g., Spam filter: Classify a spam as a non-spam
- Availability attacks: Disruption to critical services
  - Reduce the accuracy of a model

ML ATTACKER GOAL

- Confidentiality attacks: Exposure of sensitive data
  - Infer a sensitive label for a data point (e.g., hospital record)
- Integrity attacks: Unauthorized modification of data
  - Induce a model to misclassify data points from one class to another
  - e.g., Spam filter: Classify a spam as a non-spam
- Availability attacks: Disruption to critical services
  - Reduce the accuracy of a model
  - Induce a model to misclassify many data points

ATTACKER CAPABILITY



- Knowledge: Does the attacker have access to the model?
  - Training data? Learning algorithm used? Parameters?
- Attacker actions:
  - Training time: **Poisoning attacks**
  - Inference time: **Evasion attacks, model inversion attacks**

Understanding Machine Learning, Bhogavalli (2019)

POISONING ATTACKS: AVAILABILITY

POISONING ATTACKS: AVAILABILITY

- Availability: Inject mislabeled training data to damage model quality
  - 3% poisoning => 11% decrease in accuracy (Steinhardt, 2017)

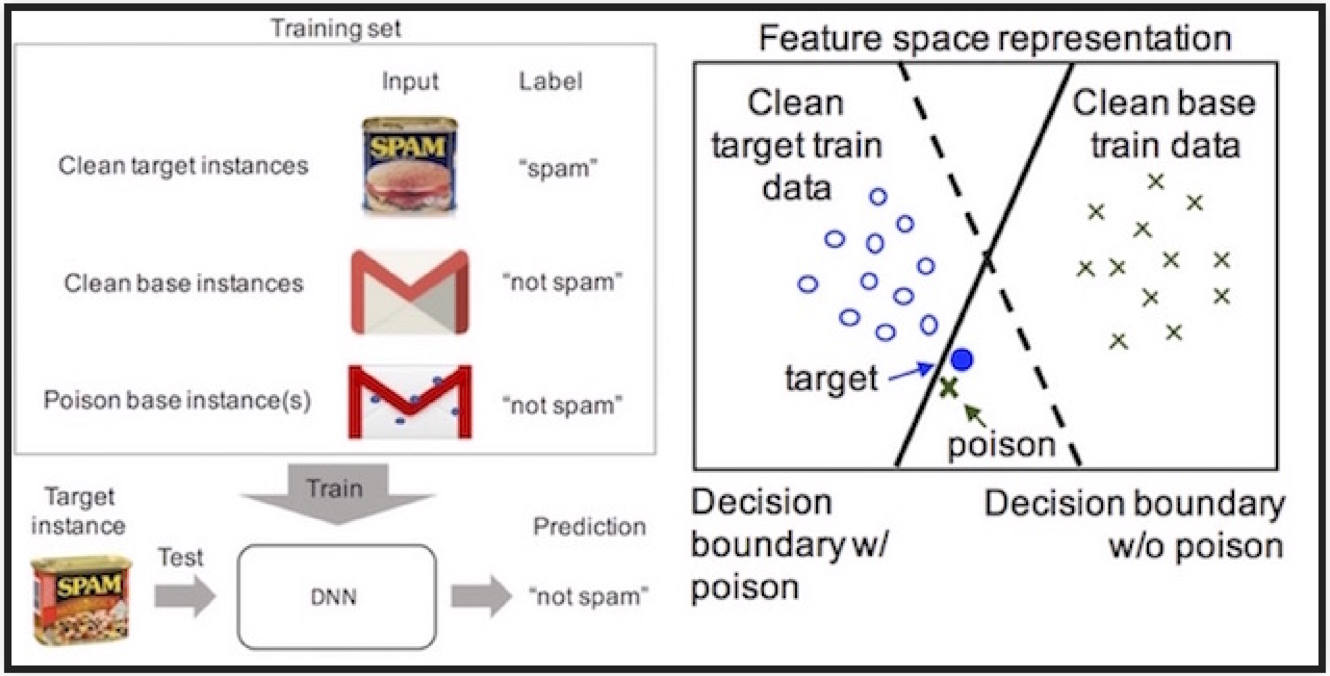
POISONING ATTACKS: AVAILABILITY

- Availability: Inject mislabeled training data to damage model quality
  - 3% poisoning => 11% decrease in accuracy (Steinhardt, 2017)
- Attacker must have some access to the training set
  - e.g., models trained on public data set (e.g., ImageNet)

POISONING ATTACKS: AVAILABILITY

- Availability: Inject mislabeled training data to damage model quality
  - 3% poisoning => 11% decrease in accuracy (Steinhardt, 2017)
- Attacker must have some access to the training set
  - e.g., models trained on public data set (e.g., ImageNet)
- Example: Anti-virus (AV) scanner
  - Online platform for submission of potentially malicious code
  - Some AV company (allegedly) poisoned competitor's model

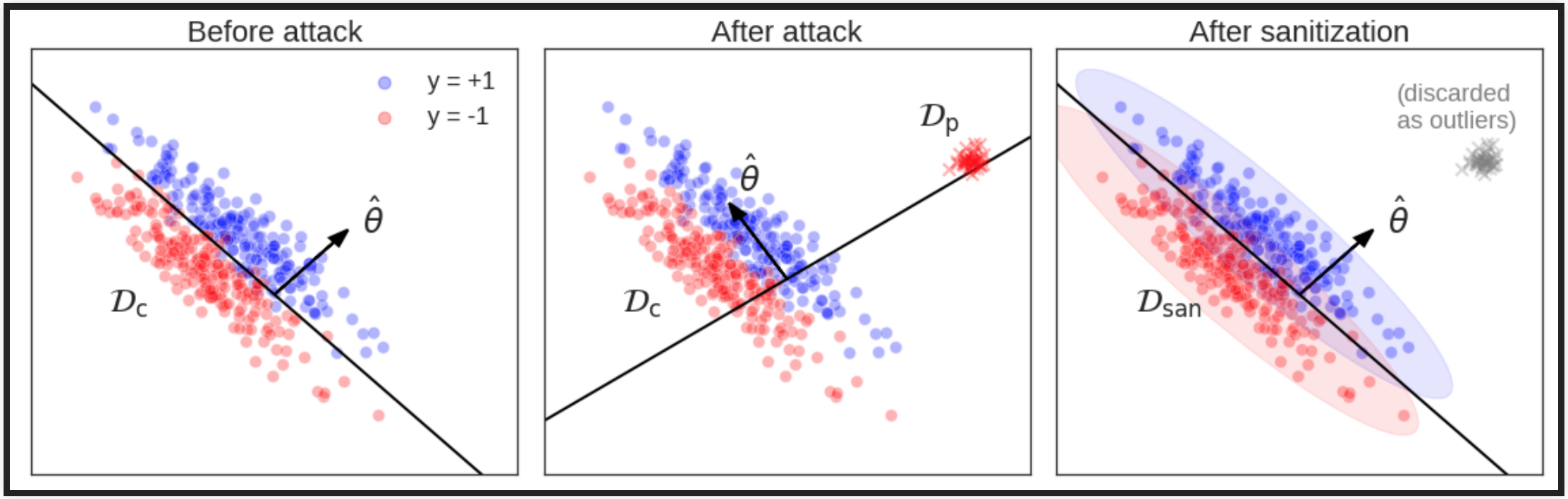
POISONING ATTACKS: INTEGRITY



- Insert training data with seemingly correct labels
- More targeted than availability attacks
    - Cause misclassification from one specific class to another

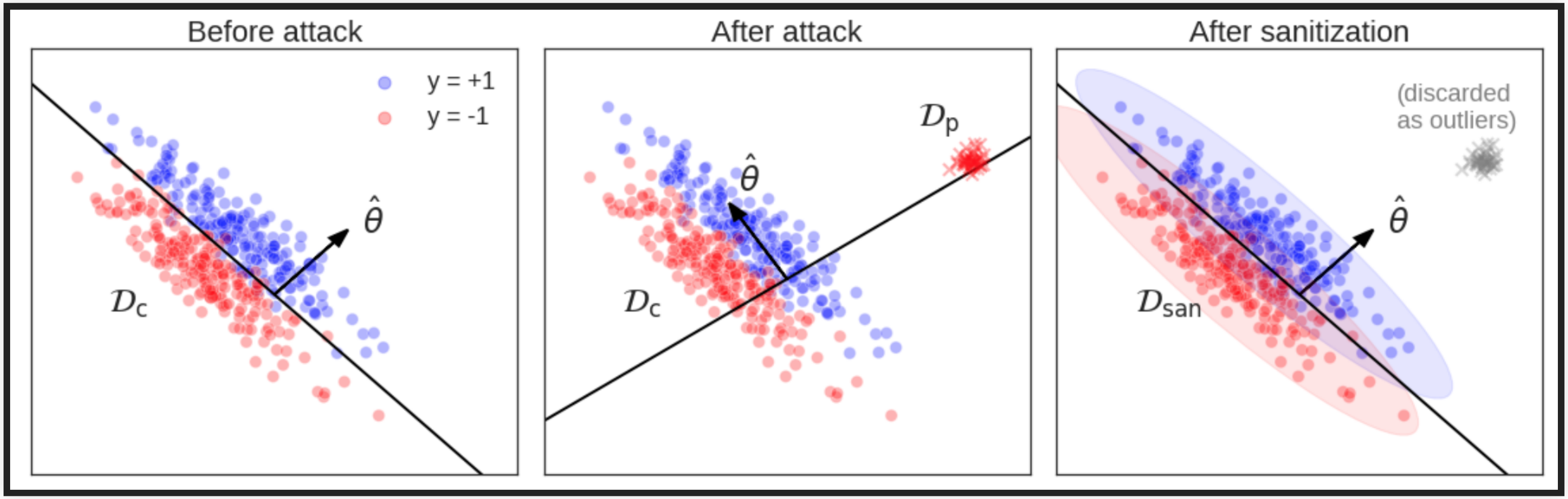
Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks, Shafahi et al. (2018)

DEFENSE AGAINST POISONING ATTACKS



Stronger Data Poisoning Attacks Break Data Sanitization Defenses, Koh, Steinhardt, and Liang (2018).

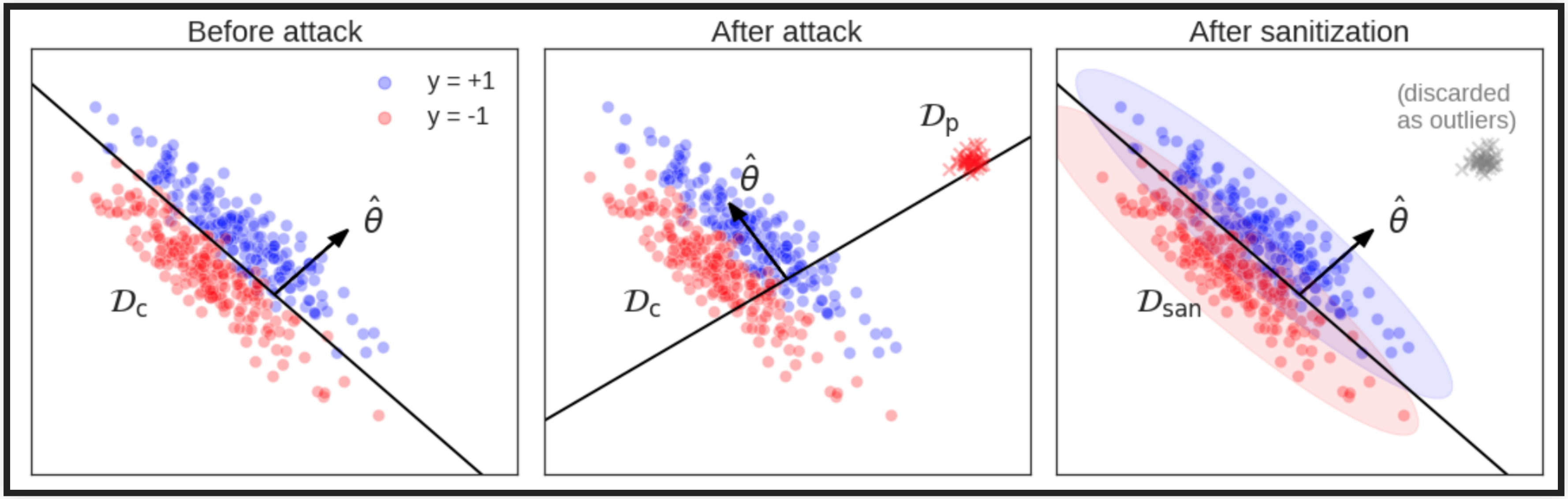
DEFENSE AGAINST POISONING ATTACKS



- Anomaly detection & data sanitization
  - Identify and remove outliers in training set (see data quality lecture)
  - Identify and understand drift from telemetry

Stronger Data Poisoning Attacks Break Data Sanitization Defenses, Koh, Steinhardt, and Liang (2018).

DEFENSE AGAINST POISONING ATTACKS



- Anomaly detection & data sanitization
  - Identify and remove outliers in training set (see data quality lecture)
  - Identify and understand drift from telemetry
- Quality control over your training data
  - Who can modify or add to my training set? Do I trust the data source?
  - Use security mechanisms (e.g., authentication) and logging to track data provenance

Stronger Data Poisoning Attacks Break Data Sanitization Defenses, Koh, Steinhardt, and Liang (2018).

EVASION ATTACKS (ADVERSARIAL EXAMPLES)

Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition, Sharif et al. (2016).

EVASION ATTACKS (ADVERSARIAL EXAMPLES)

- Add noise to an existing sample & cause misclassification

Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition, Sharif et al. (2016).

EVASION ATTACKS (ADVERSARIAL EXAMPLES)

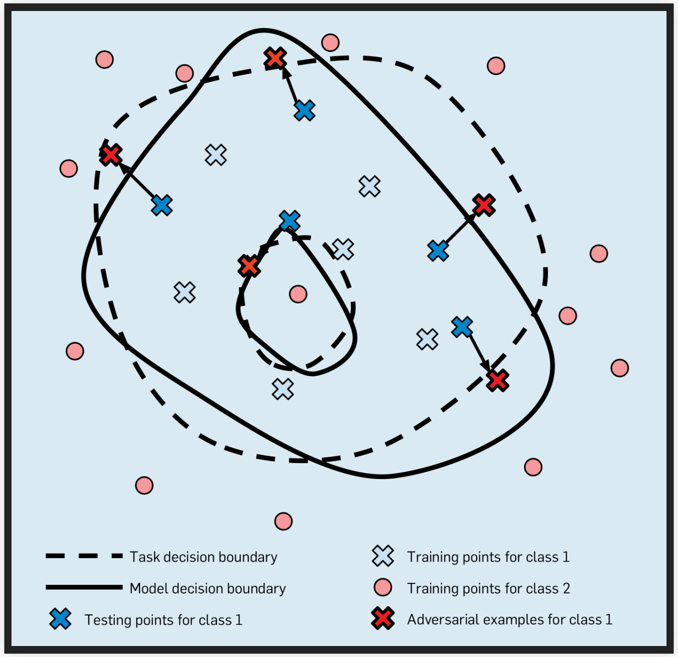
- Add noise to an existing sample & cause misclassification
- Attack at inference time
    - Typically assumes knowledge of the model (algorithm, parameters)
    - Recently, shown to be possible even when the attacker only has access to model output ("blackbox" attack)

Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition, Sharif et al. (2016).

EVASION ATTACKS: ANOTHER EXAMPLE

Robust Physical-World Attacks on Deep Learning Visual Classification, Eykholt et al., in CVPR (2018).

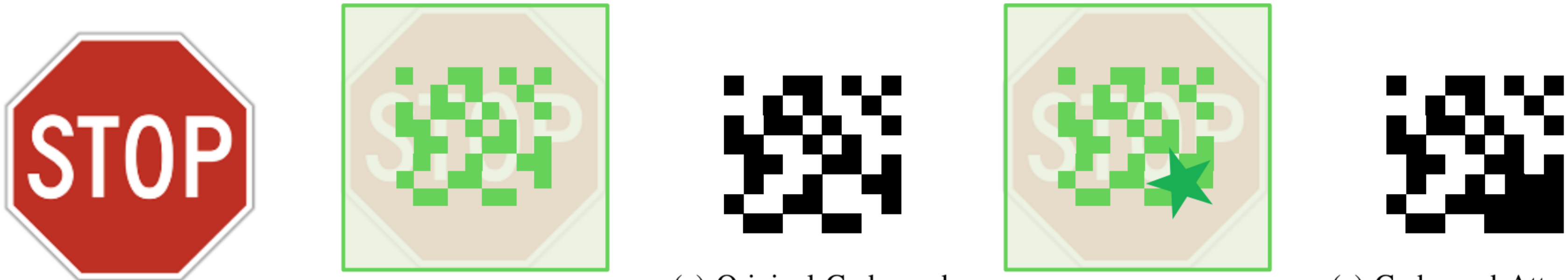
TASK DECISION BOUNDARY VS MODEL BOUNDARY



- Decision boundary: Ground truth; often unknown and not specifiable
- Model boundary: What the model learns; an approximation of decision boundary

From Goodfellow et al (2018). Making machine learning robust against adversarial inputs. *Communications of the ACM*, *61*(7), 56-66.

DEFENSE AGAINST EVASION ATTACKS



(a) Visual Image

(b) Infrared Image of Smart Code

(c) Original Codeword

(d) Infrared Image of Smart Code Attacked

(e) Codeword Attacked

Reliable Smart Road Signs, Sayin et al. (2019).

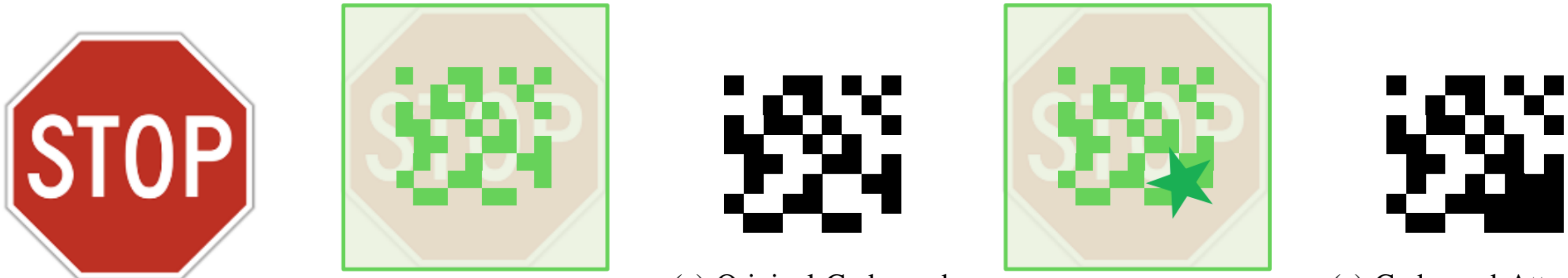
DEFENSE AGAINST EVASION ATTACKS

(a) Visual Image (b) Infrared Image of Smart Code (c) Original Codeword (d) Infrared Image of Smart Code Attacked (e) Codeword Attacked

- Adversarial training
  - Generate/find a set of adversarial examples
  - Re-train your model with correct labels

Reliable Smart Road Signs, Sayin et al. (2019).

DEFENSE AGAINST EVASION ATTACKS

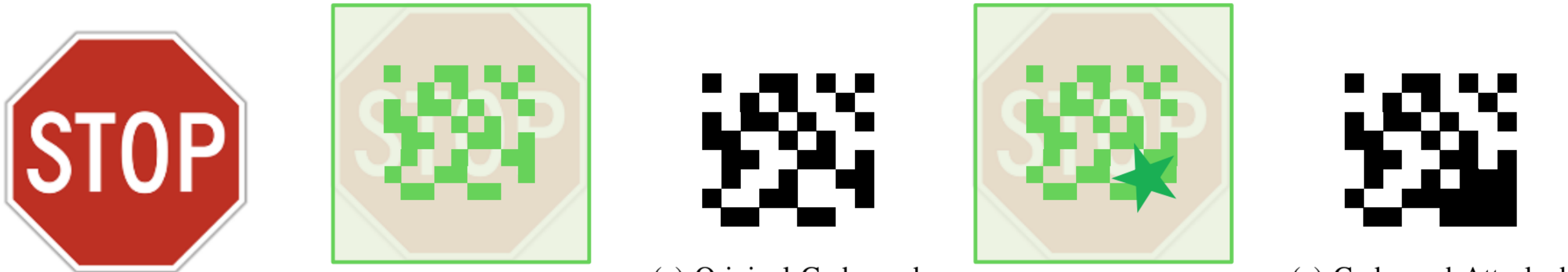


(a) Visual Image (b) Infrared Image of Smart Code (c) Original Codeword (d) Infrared Image of Smart Code Attacked (e) Codeword Attacked

- Adversarial training
  - Generate/find a set of adversarial examples
  - Re-train your model with correct labels
- Input sanitization
  - "Clean" & remove noise from input samples
  - e.g., Color depth reduction, spatial smoothing, JPEG compression

Reliable Smart Road Signs, Sayin et al. (2019).

DEFENSE AGAINST EVASION ATTACKS



(a) Visual Image (b) Infrared Image of Smart Code (c) Original Codeword (d) Infrared Image of Smart Code Attacked (e) Codeword Attacked

- Adversarial training
    - Generate/find a set of adversarial examples
    - Re-train your model with correct labels
- Input sanitization
    - "Clean" & remove noise from input samples
    - e.g., Color depth reduction, spatial smoothing, JPEG compression
- Redundancy: Design multiple mechanisms to detect an attack
    - Stop sign: Insert a barcode as a checksum; harder to bypass

Reliable Smart Road Signs, Sayin et al. (2019).

GENERATING ADVERSARIAL EXAMPLES

- Q. How do we generate adversarial examples?

GENERATING ADVERSARIAL EXAMPLES

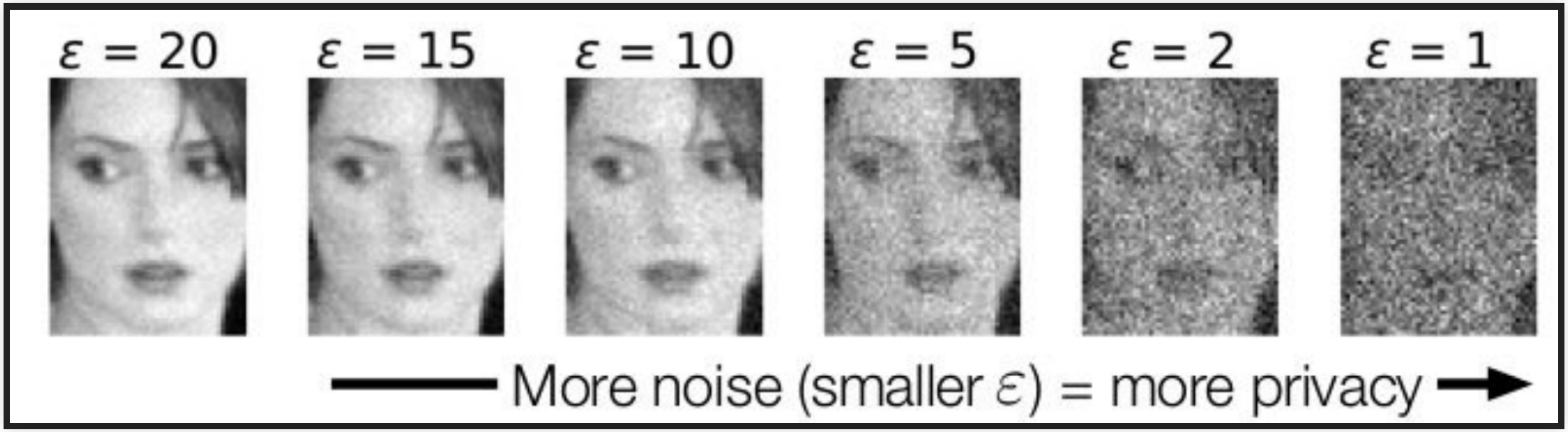
- See counterfactual explanations
- Find similar input with different prediction
  - targeted (specific prediction) vs untargeted (any wrong prediction)
- Many similarity measures (e.g., change one feature vs small changes to many features)
  - $x^* = x + argmin\{ |\epsilon| : f(x + \epsilon) \neq f(x)\}$
- Attacks more effective with access to model internals, but also black-box attacks (with many queries to the model) feasible
  - With model internals: follow the model's gradient
  - Without model internals: learn surrogate model
  - With access to confidence scores: heuristic search (e.g., hill climbing)

MODEL INVERSION: CONFIDENTIALITY

- Given a model output (e.g., name of a person), infer the corresponding, potentially sensitive input (facial image of the person)
- One method: Gradient descent on input space
    - Assumes that the model produces a confidence score for prediction
    - Start with a random input vector & iterate towards input values with higher confidence level

Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures, M. Fredrikson et al. in CCS (2015).

DEFENSE AGAINST MODEL INVERSION ATTACKS



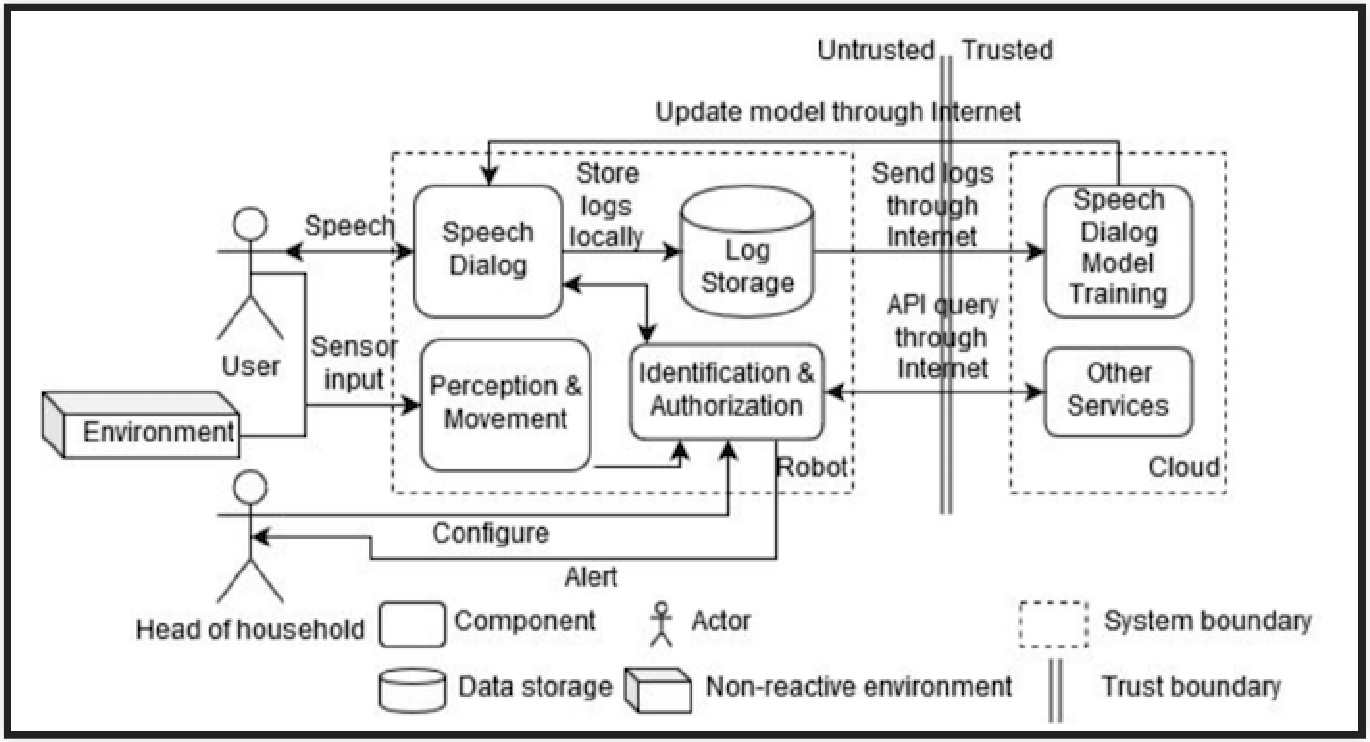
- Limit attacker access to confidence scores
  - e.g., reduce the precision of the scores by rounding them off
  - But also reduces the utility of legitimate use of these scores!
- Differential privacy in ML
  - Limit what attacker can learn about the model (e.g., parameters) based on an individual training sample
  - Achieved by adding noise to input or output (e.g., DP-SGD)
  - More noise => higher privacy, but also lower model accuracy!

Biscotti: A Ledger for Private and Secure Peer-to-Peer Machine Learning, M. Shayan et al., arXiv:1811.09904 (2018).

BREAKOUT: HOME ASSISTANT ROBOT

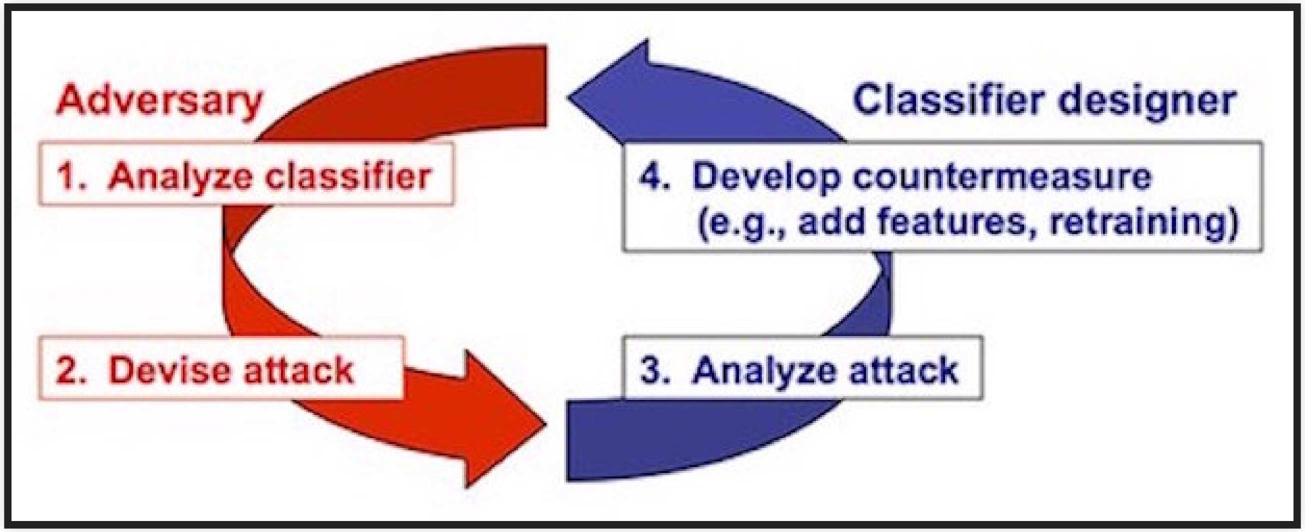
- Dialogue system to interact with family members
- Use perception & speech to identify the person
- Notify owner if stranger detected in the house
- Log & upload interactions; re-train & update models for all robots

BREAKOUT: HOME ASSISTANT ROBOT

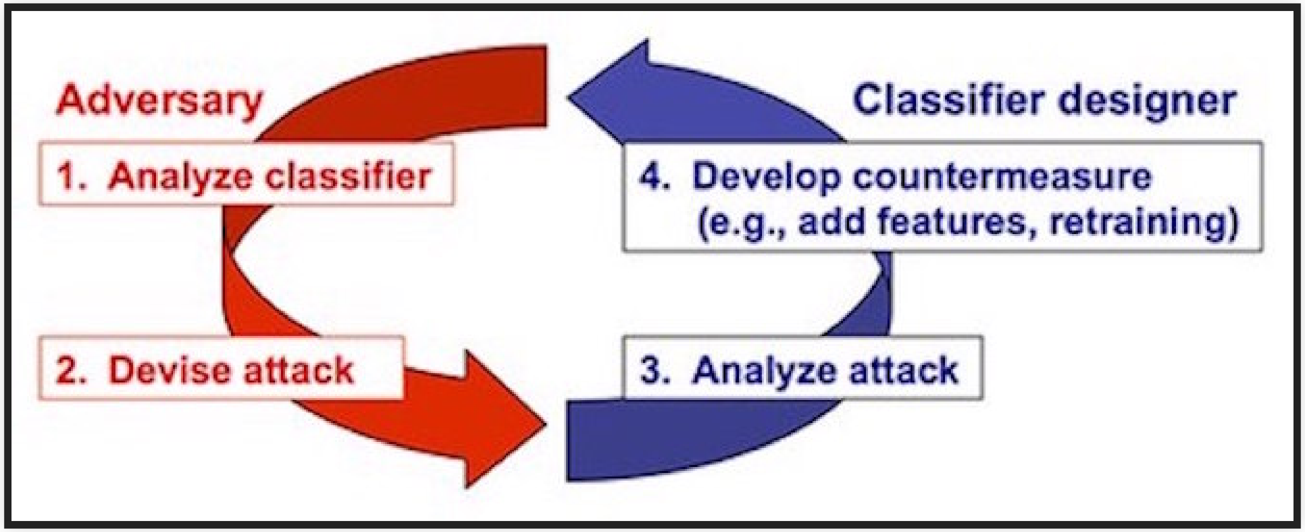


- What are the security requirements?
- What are possible attacks on the system?
- How can we defend against some of them?

STATE OF ML SECURITY

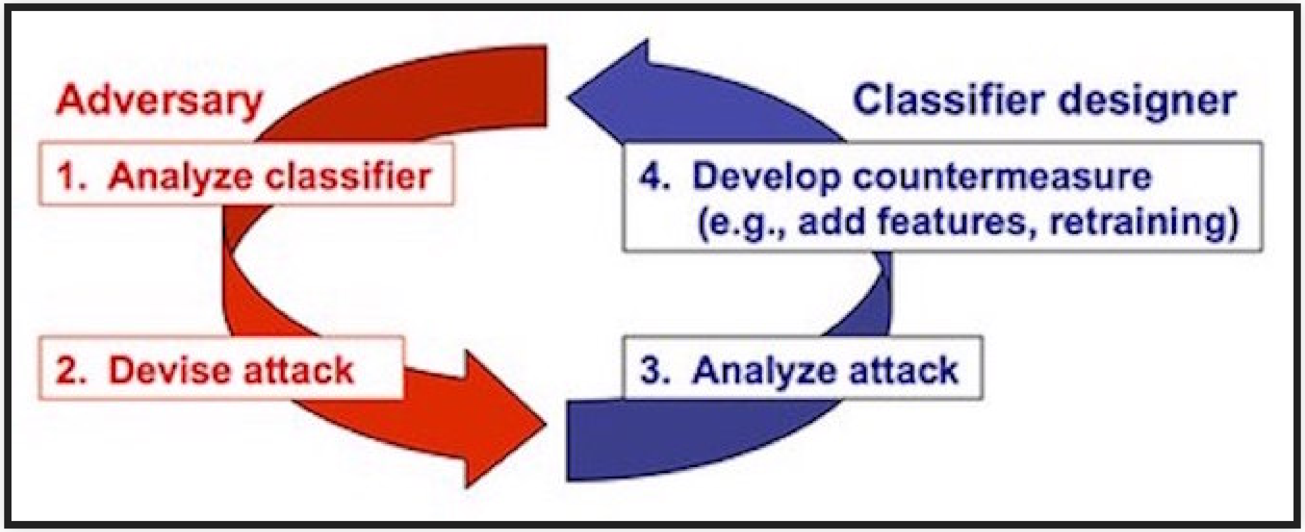


STATE OF ML SECURITY



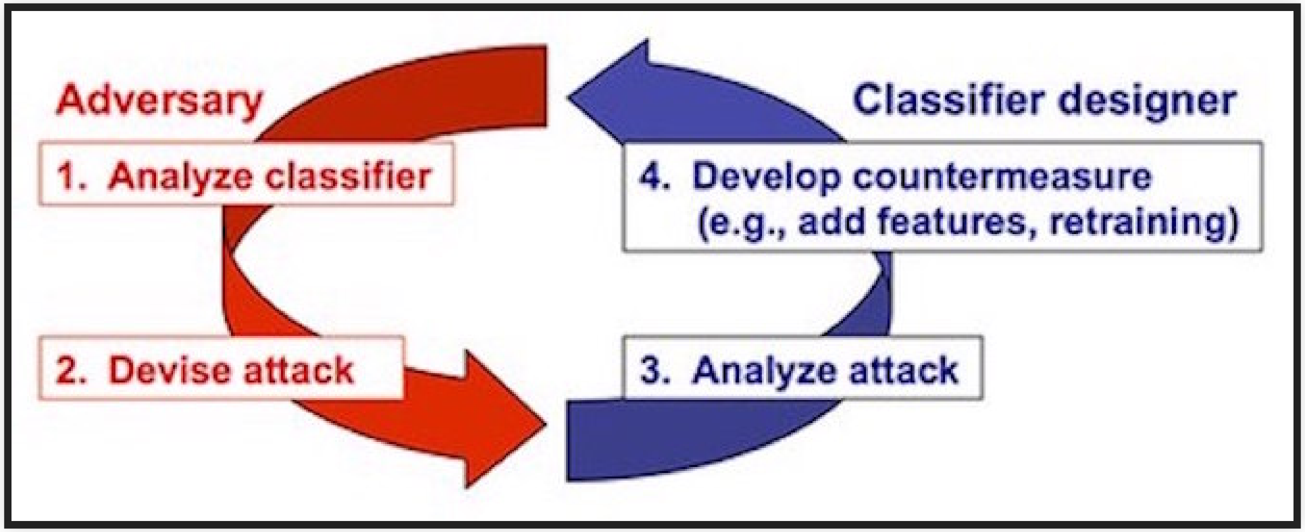
- On-going arms race (mostly among researchers)
  - Defenses proposed & quickly broken by noble attacks

STATE OF ML SECURITY



- On-going arms race (mostly among researchers)
  - Defenses proposed & quickly broken by noble attacks
- Assume ML component is likely vulnerable
  - Design your system to minimize impact of an attack

STATE OF ML SECURITY



- On-going arms race (mostly among researchers)
  - Defenses proposed & quickly broken by noble attacks
- Assume ML component is likely vulnerable
  - Design your system to minimize impact of an attack
- Remember: There may be easier ways to compromise system
  - e.g., poor security misconfiguration (default password), lack of encryption, code vulnerabilities, etc.,

DESIGNING FOR SECURITY

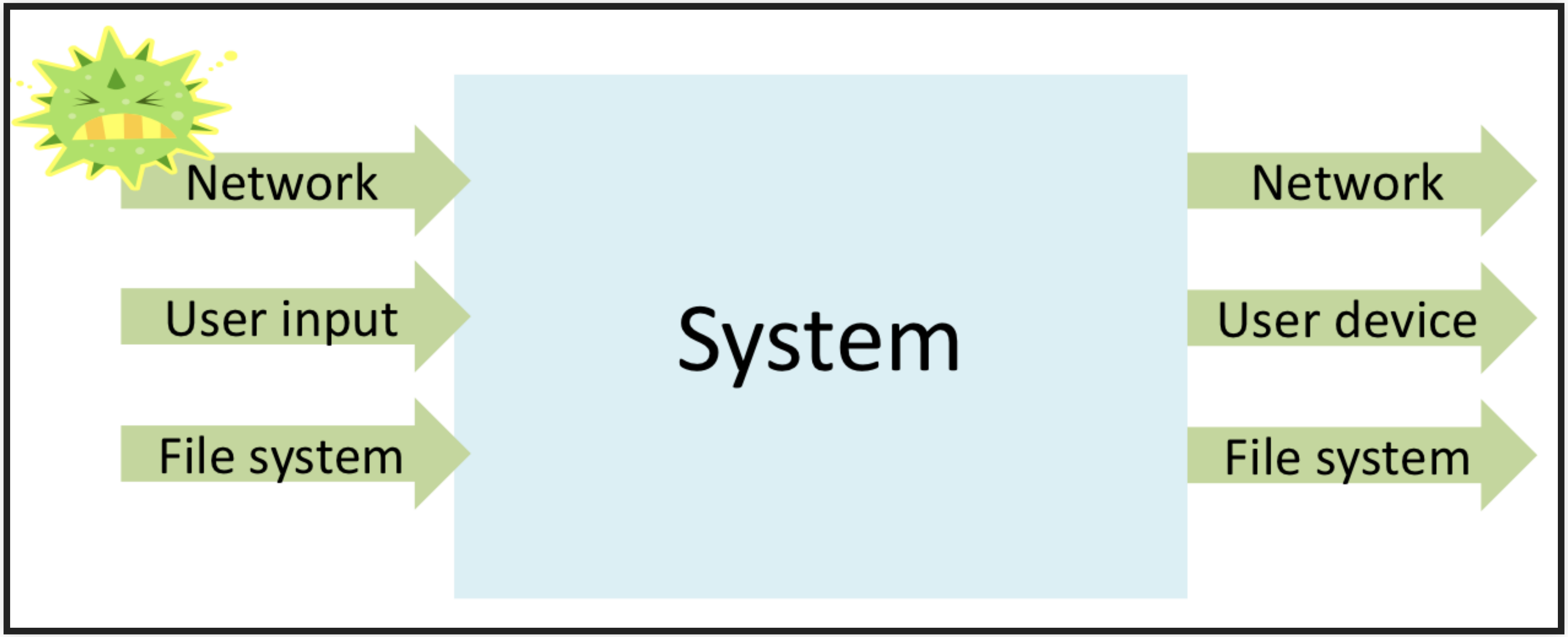
SECURITY MINDSET

- Assume that all components may be compromised at one point or another
- Don't assume users will behave as expected; assume all inputs to the system as potentially malicious
- Aim for risk minimization, not perfect security; reduce the chance of catastrophic failures from attacks

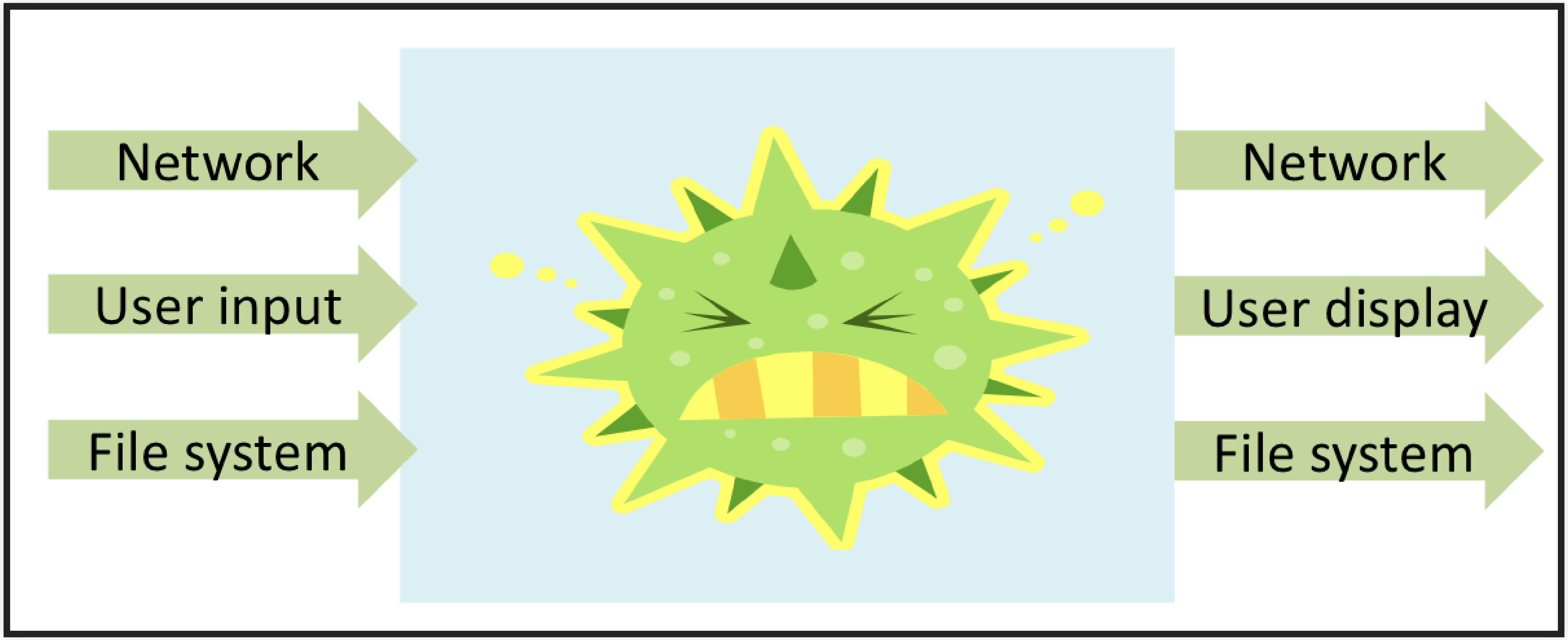
SECURE DESIGN PRINCIPLES

- Principle of least privilege
    - A component should be given the minimal privileges needed to fulfill its functionality
- Isolation/compartmentalization
    - Components should be able to interact with each other no more than necessary
    - Components should treat inputs from each other as potentially malicious
- Goal: Minimize the impact of a compromised component on the rest of the system
    - In poor system designs, vulnerability in one component => entire system compromised!

MONOLITHIC DESIGN

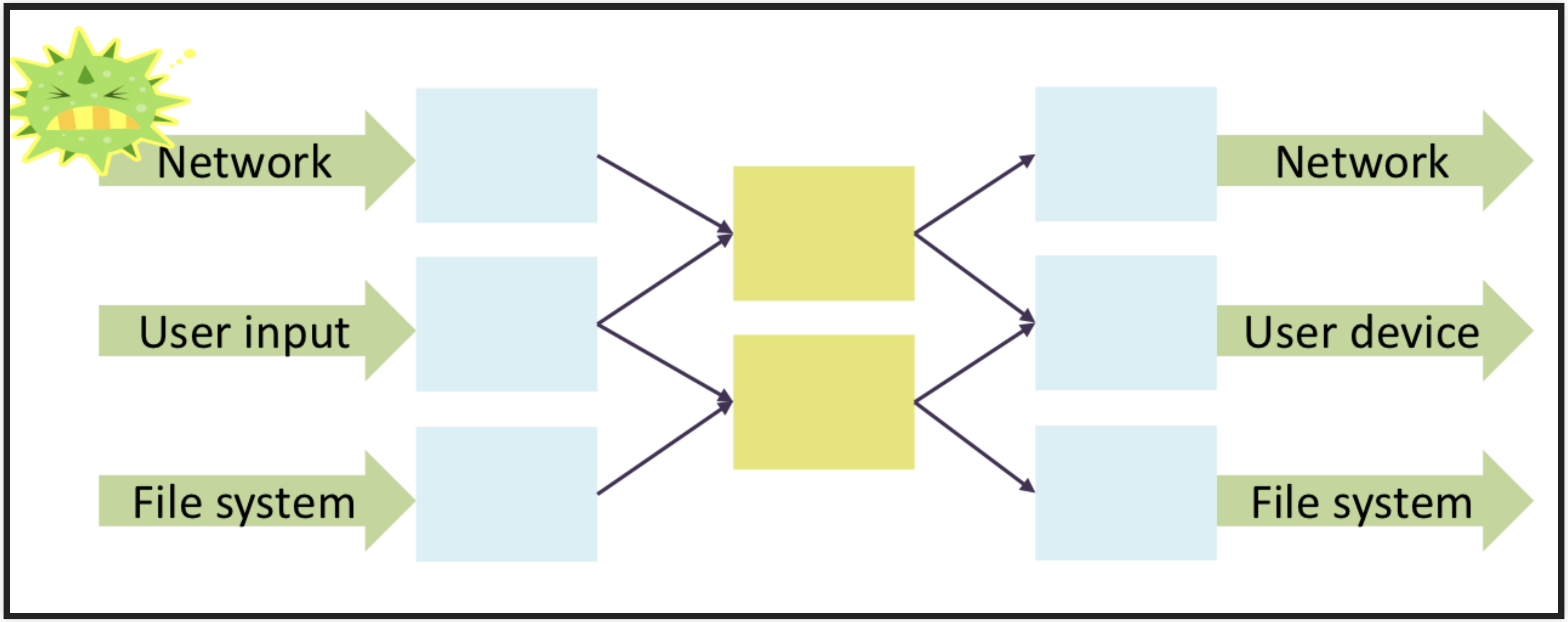


MONOLITHIC DESIGN

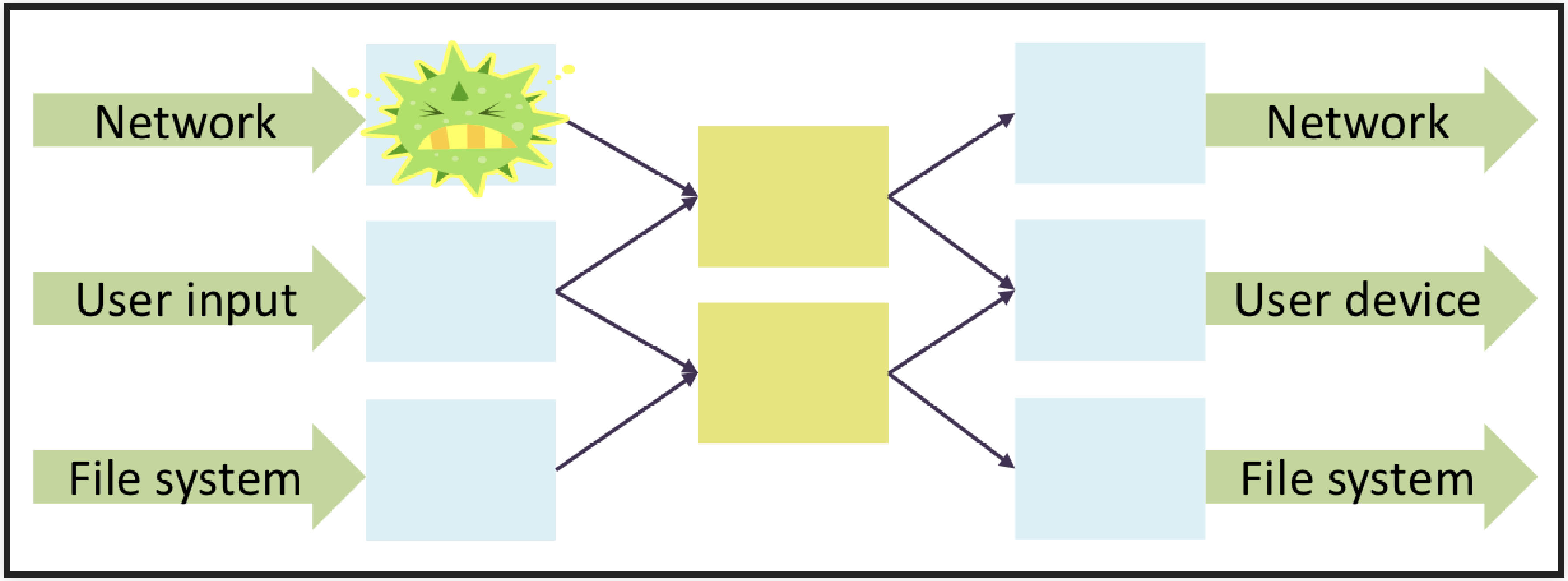


Flaw in any part of the system => Security impact on the entire system!

COMPARTMENTALIZED DESIGN

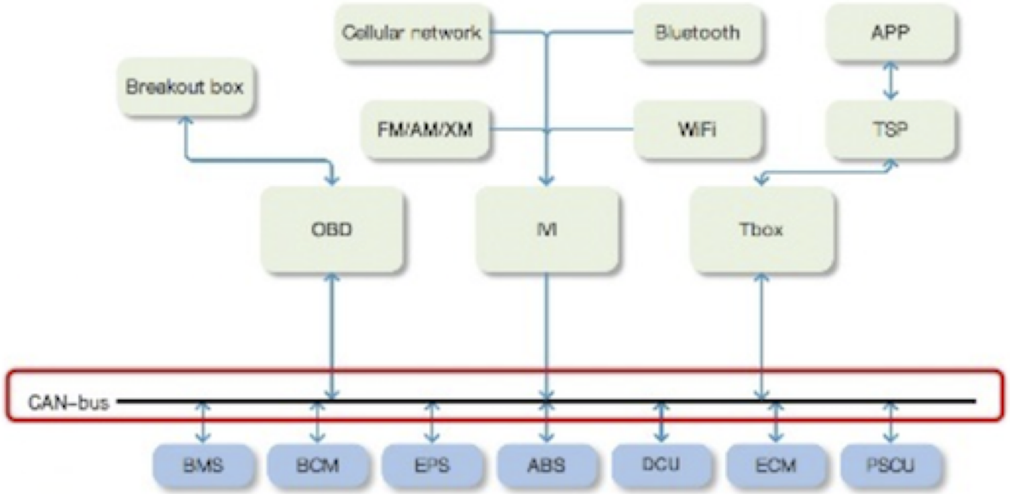


COMPARTMENTALIZED DESIGN



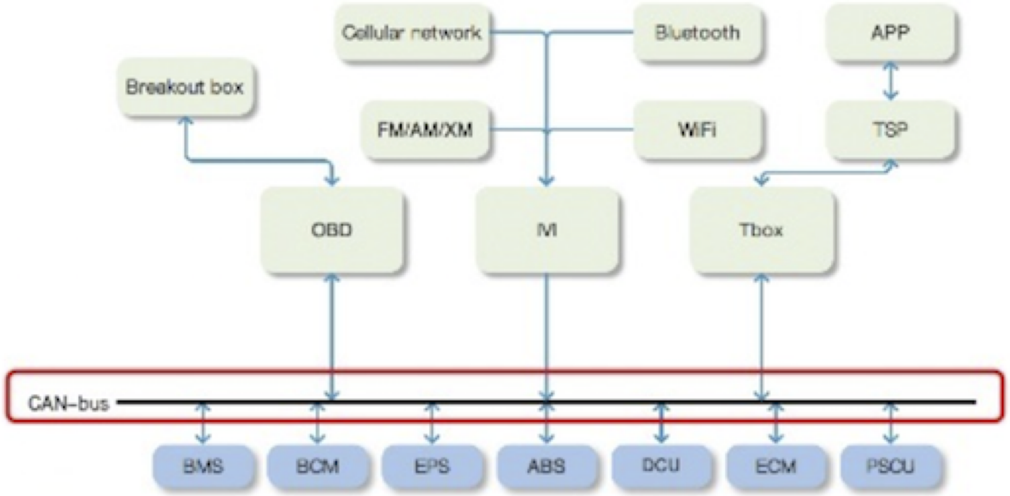
Flaw in one component => Limited impact on the rest of the system!

EXAMPLE: VEHICLE SECURITY



- Research project@UCSD: Remotely taking over vehicle control
    - Create MP3 with malicious code & burn onto CD
    - Play CD => send malicious commands to brakes, engine, locks...
- Problem: Over-privilege & lack of isolation!
    - In traditional vehicles, components share a common CAN bus
    - Anyone can broadcast & read messages

EXAMPLE: VEHICLE SECURITY

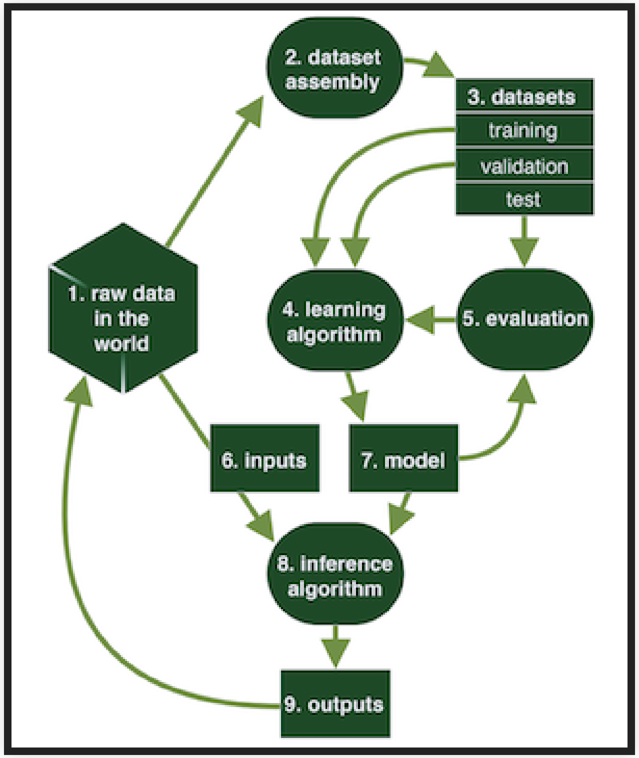


- Research project@UCSD: Remotely taking over vehicle control
    - Create MP3 with malicious code & burn onto CD
    - Play CD => send malicious commands to brakes, engine, locks...
- Problem: Over-privilege & lack of isolation!
    - In traditional vehicles, components share a common CAN bus
    - Anyone can broadcast & read messages

Comprehensive Experimental Analyses of Automotive Attack Surfaces, Checkoway et al., in USENIX Security (2011).

SECURE DESIGN PRINCIPLES FOR ML

- Principle of least privilege
  - Who has access to training data, model internal, system input & output, etc.,?
  - Does any user/stakeholder have more access than necessary?
    - If so, limit access by using authentication mechanisms



SECURE DESIGN PRINCIPLES FOR ML

- Principle of least privilege
    - Who has access to training data, model internal, system input & output, etc.,?
    - Does any user/stakeholder have more access than necessary?
        - If so, limit access by using authentication mechanisms

SECURE DESIGN PRINCIPLES FOR ML

- Principle of least privilege
  - Who has access to training data, model internal, system input & output, etc.,?
  - Does any user/stakeholder have more access than necessary?
    - If so, limit access by using authentication mechanisms
- Isolation & compartmentalization
  - Can a security attack on one ML component (e.g., misclassification) adversely affect other parts of the system?
    - If so, compartmentalize or build in mechanisms to limit impact (see risk mitigation strategies)

SECURE DESIGN PRINCIPLES FOR ML

- Principle of least privilege
  - Who has access to training data, model internal, system input & output, etc.,?
  - Does any user/stakeholder have more access than necessary?
    - If so, limit access by using authentication mechanisms
- Isolation & compartmentalization
  - Can a security attack on one ML component (e.g., misclassification) adversely affect other parts of the system?
    - If so, compartmentalize or build in mechanisms to limit impact (see risk mitigation strategies)
- Monitoring & detection:
  - Look for odd shifts in the dataset and clean the data if needed (for poisoning attacks)
  - Assume all system input as potentially malicious & sanitize (evasion attacks)

AI FOR SECURITY

built in

TECH JOBS

30 COMPANIES MERGING AI AND CYBERSECURITY TO KEEP US SAFE AND SOUND

Alyssa Schroer

July 12, 2019 Updated: July 15, 2020

By the year 2021, cybercrime losses will

MANY DEFENSE SYSTEMS USE MACHINE LEARNING

- Classifiers to learn malicious content
  - Spam filters, virus detection
- Anomaly detection
  - Identify unusual/suspicious activity, eg. credit card fraud, intrusion detection
  - Often unsupervised learning, e.g. clustering
- Game theory
  - Model attacker costs and reactions, design countermeasures
- Automate incidence response and mitigation activities
  - Integrated with DevOps
- Network analysis
  - Identify bad actors and their communication in public/intelligence data
- Many more, huge commercial interest

Recommended reading: Chandola, Varun, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey." ACM computing surveys (CSUR) 41, no. 3 (2009): 1-58.

AI SECURITY SOLUTIONS ARE AI-ENABLED SYSTEMS TOO

- AI/ML component one part of a larger system
- Consider entire system, from training to telemetry, to user interface, to pipeline automation, to monitoring
- AI-based security solutions can be attacked themselves

EQUIFAX

Speaker notes

One contributing factor to the Equifax attack was an expired certificate for an intrusion detection system

SUMMARY

- Security requirements: Confidentiality, integrity, availability
- Threat modeling to identify security requirements & attacker capabilities
- ML-specific attacks on training data, telemetry, or the model
  - Poisoning attack on training data to influence predictions
  - Evasion attacks to shape input data to achieve intended predictions (adversarial learning)
  - Model inversion attacks for privacy violations
- Security design at the system level
  - Principle of least privilege
  - Isolation & compartmentalization
- AI can be used for defense (e.g. anomaly detection)
- **Key takeaway**: Adopt a security mindset! Assume all components may be vulnerable in one way or another. Design your system to explicitly reduce the impact of potential attacks