A New Doctrine for Hardware Security

Adam Hastings, Ryan Piersma and **Prof. Simha Sethumadhavan**
Columbia University

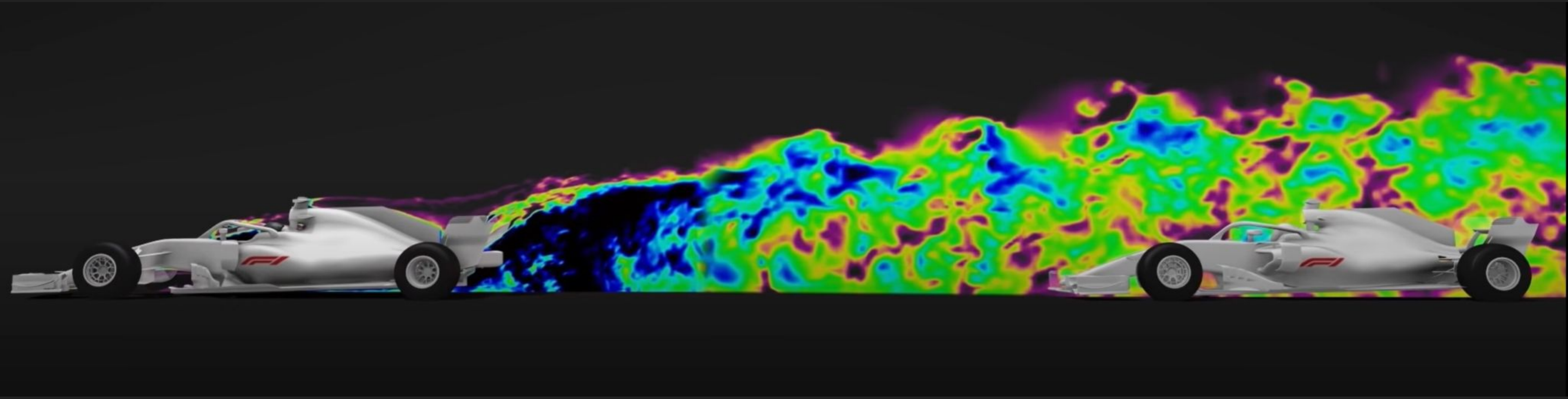
Software and Supply Chain Assurance Forum

March 2nd 2022

simha@columbia.edu
https://linkedin.com/in/simha
@TheSimha

https://arxiv.org/abs/2203.05015
https://arxiv.org/abs/2007.09537


PETRONAS
SYNTIUM
AMG



Game:

Build the fastest rule-abiding car

Strategy:

Win races without overtaking

Outcome:

Races are uneventful (and boring)


2022 CAR
2022
CLEAN AIR

Mechanism Design:

Design the rules of a game such that players will choose strategies that lead to some desired overall outcome

This Talk:

A New Mechanism Design for (Hardware) Security

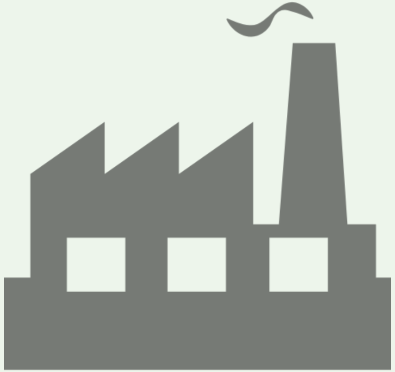
(based on new way of thinking about security)

Defenders keep losing.
Change the game.

Who is to Blame for Security Failures?

How do we change behaviors?

Attackers?



Vendors?

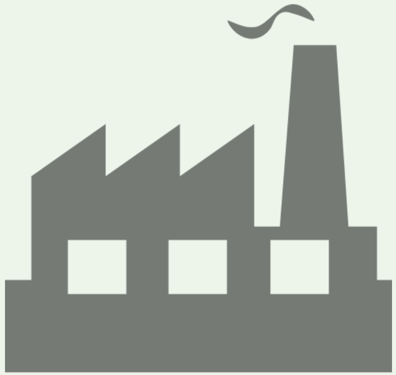
Users?

Authorities?

Is there a single point intervention to bring about desired change?

How do we change behaviors?

Attackers?



Vendors?

Users?

Authorities?

How do we change behaviors?

The Doctrine of Shared Burdens

The burden of security should be borne **equitably** between the Users, Vendors, Authorities and Attackers.

Att

ties?

Three Case Studies

(to illustrate the complexities of behavioral change)

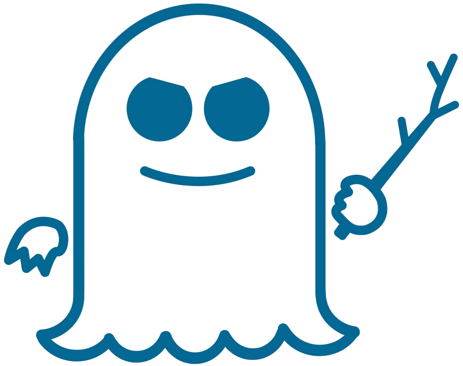
Case Study #1: Spectre

- Modern processors speculatively execute instructions to improve performance. Significant performance gains (~1.5-3x)

- Problem: During speculative execution, transient instructions can perform actions not intended by the programmer

- Who should "pay" for this?
    - Processor vendors
    - Programmers who write critical code
    - End users who care about security

- Vendors won't fix Spectre-v1



Case Study #1: Spectre

- Processor vendor has two options
  - Secure by default: First mover disadvantage for vendors due to high cost
  - Two SKUs: Fast or Secure causes inequity, Hobson's choice

- Programmer can fix but
  - Burdensome (though Google Chrome does this)
  - Externalizes cost

- User can decide to turn on or turn off security as needed, but
  - Users often don't know what they need (classic information asymmetry)
  - Externalizes risk and cost

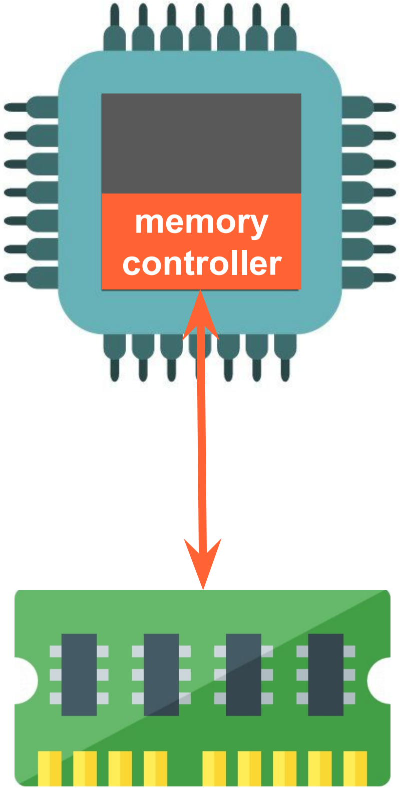
- Need new mechanisms to resolve this moral hazard

Case Study #2: Rowhammer

- DRAM cells are so small that their bits can be flipped by repeating activating nearby memory

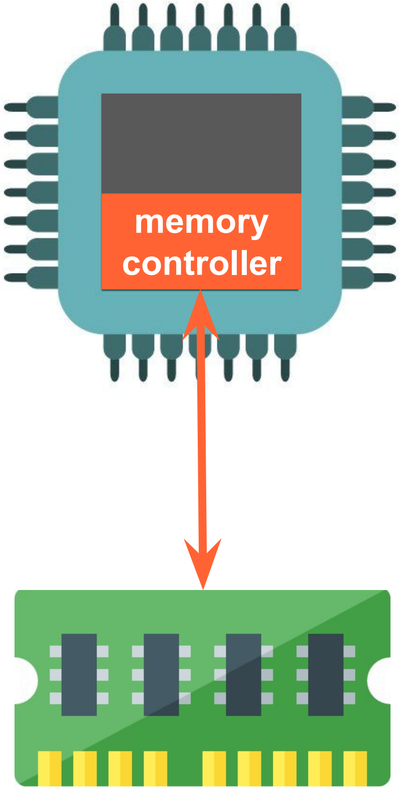
- Problem: Many stakeholders. Who should fix?
    - DRAM vendors?
    - Memory controller manufacturers?
    - Processor/SoC integrators?
    - Programmers?
    - End users?

- Currently
    - JEDEC co-ordinates stakeholders to create standards



Case Study #2: Rowhammer Solutions

- DRAM vendors
  - Secure by default: 1st mover disadvantage for vendors due to cost
  - Two SKUs: Secure or Cheap DRAMs (Hobson's choice)
- SoC/Memory controller IP providers
  - Solution: faster refreshes to restore state
  - Vendors product consumes more energy; moral hazard
- Programmers/Users?
  - Information asymmetry, burdensome
  - Moral hazard
- Current solution: RFM
  - It is complicated: SoC vendor and DRAM work together (JEDEC)
  - User pays a constant but small cost
  - Outcome: Security TBD
- Solution can be really simplified if DRAM vendors can be incentivized to set aside for security.



Case Study #3: Hardware Patching

- Recent CPU vulnerabilities have necessitated patches that come with a performance cost.
- Problem: Vendors could be disincentivized from releasing security patches in a timely manner
- Customers may not know about pending patches
- Need mechanisms to solve information asymmetry and adverse selection

Fixing hardware security problems requires more than technical solutions!

The Cyber Social Contract

How to Rebuild Trust in a Digital World

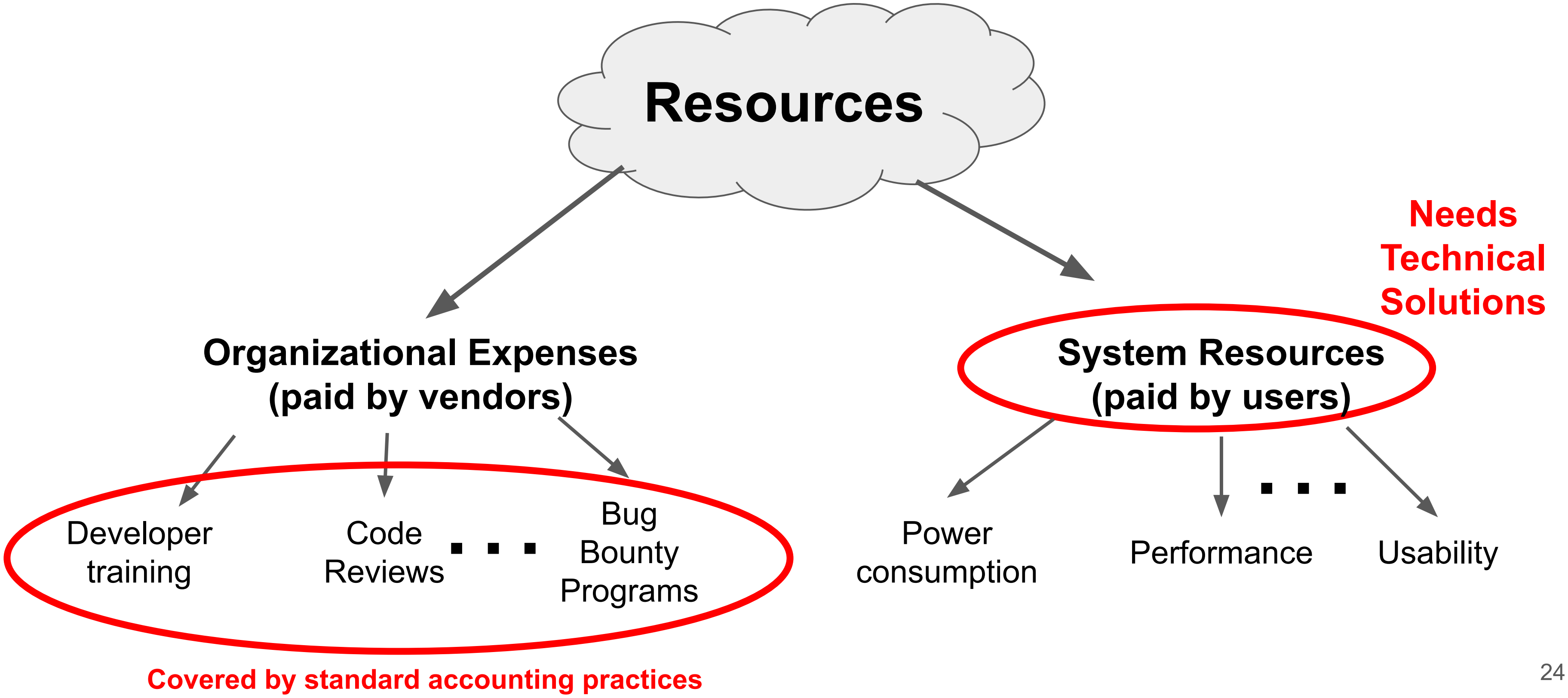
By Chris Inglis and Harry Krejsa February 21, 2022

Still, its contours are already clear: the private sector must prioritize long-term investments in a digital ecosystem that equitably distributes the burden of cyberdefense.

Open Mandates: A Novel Mechanism

Require *all* vendors to spend some percentage of their resources on security.

Spending “Resources” on Security…?



COMMAND Overview

Certifiable Open Measurable Mandate

Regulator

Vendor

End User

COMMAND Overview

Addresses first mover disadvantage

Not prescriptive!

Regulator mandates that *all* Vendors must dedicate fixed % of resources for security

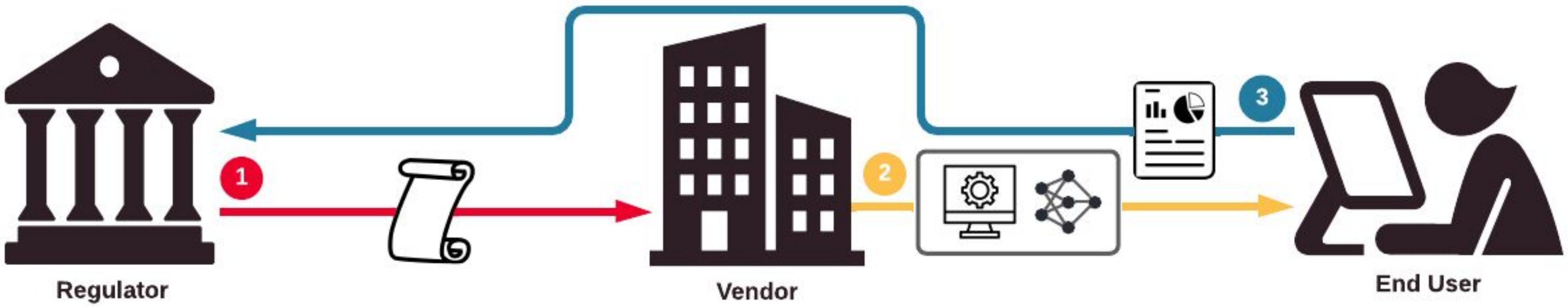
COMMAND Overview

Vendor ships products with on-device model that checks that security mandates are satisfied

Users don't have to decide which defenses to employ and solves information asymmetry

COMMAND Overview

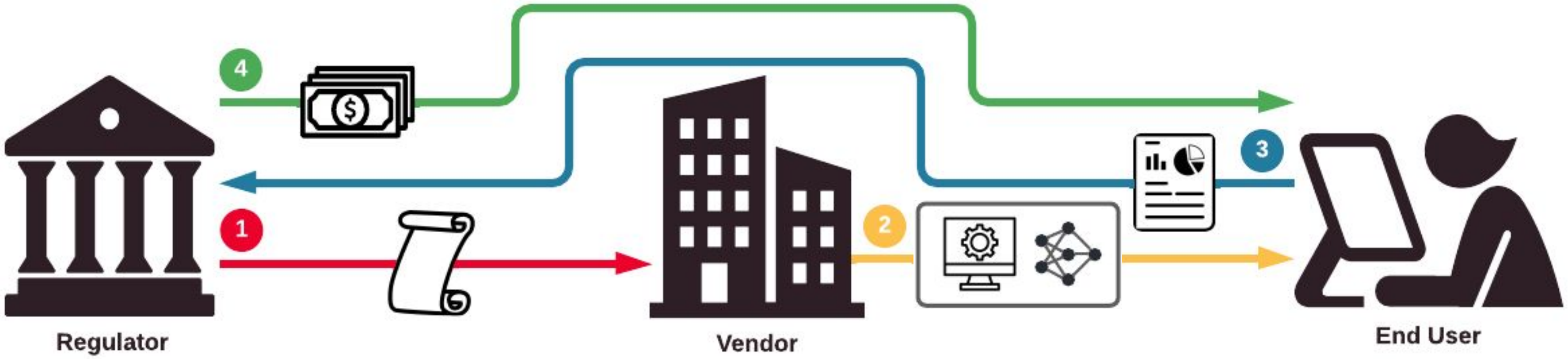
Users reports incurred overheads to Regulators



COMMAND Overview

Regulator rewards participants with tax incentives

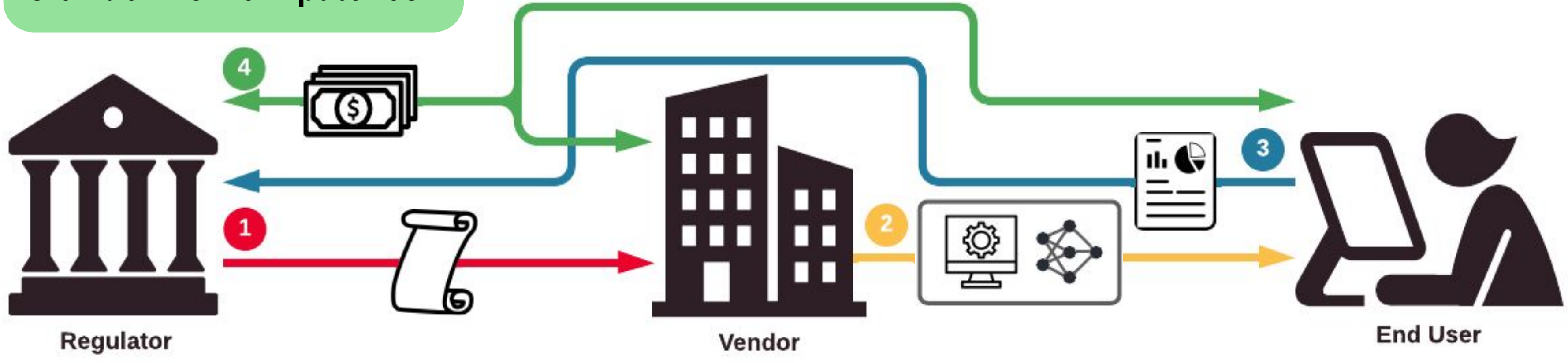
Offsets cost of security investment.



COMMAND Overview

Regulator also mediates rebates between Vendor and User for performance slowdowns from patches

Removes barriers to timely patching.



Benefits of COMMAND

Authorities

- Not prescriptive: Authorities do not have be involved in picking security solutions
- Creates an auditable paper trail of security investments
- Promotes innovation: Vendors will compete to include as many security solutions as possible within the security budget

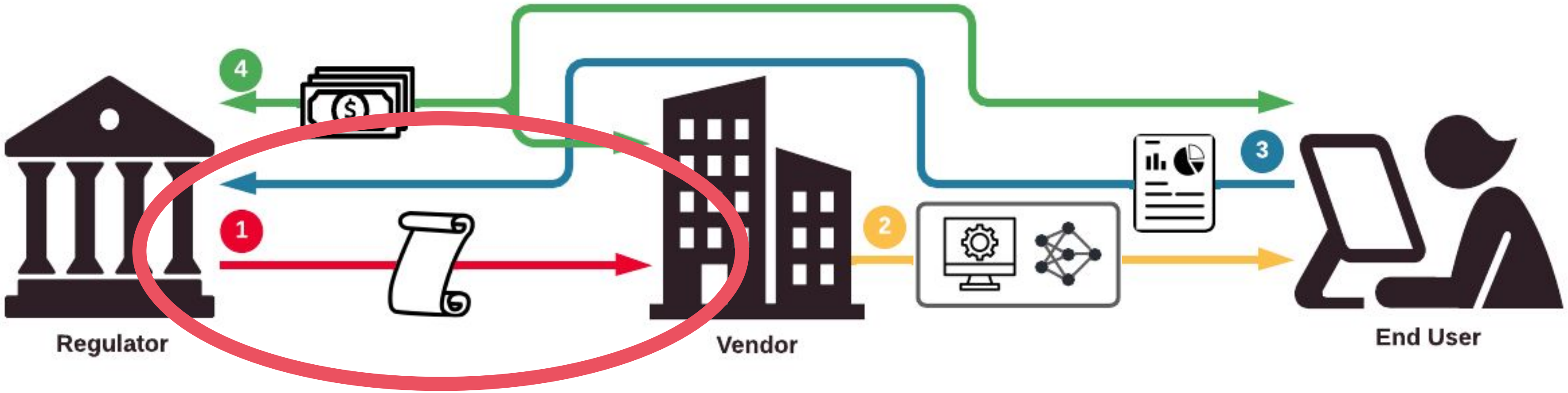
Vendors

- Avoids first mover disadvantage: all vendors have to pay a min for security
- *a la carte* discount for hardware patches *iff* they slow systems based on end user patterns

End users

- Minimizes information asymmetry
- Incentives security

COMMAND Overview



Can Open Mandates Work?

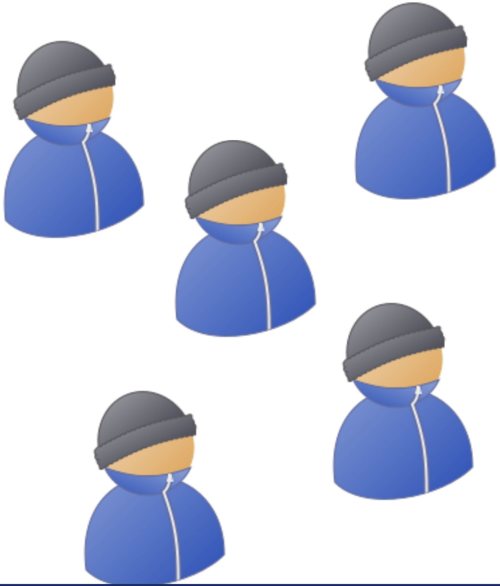
An Attacker vs. Defender Simulation

- Goal: understand the dynamics between attackers and defenders

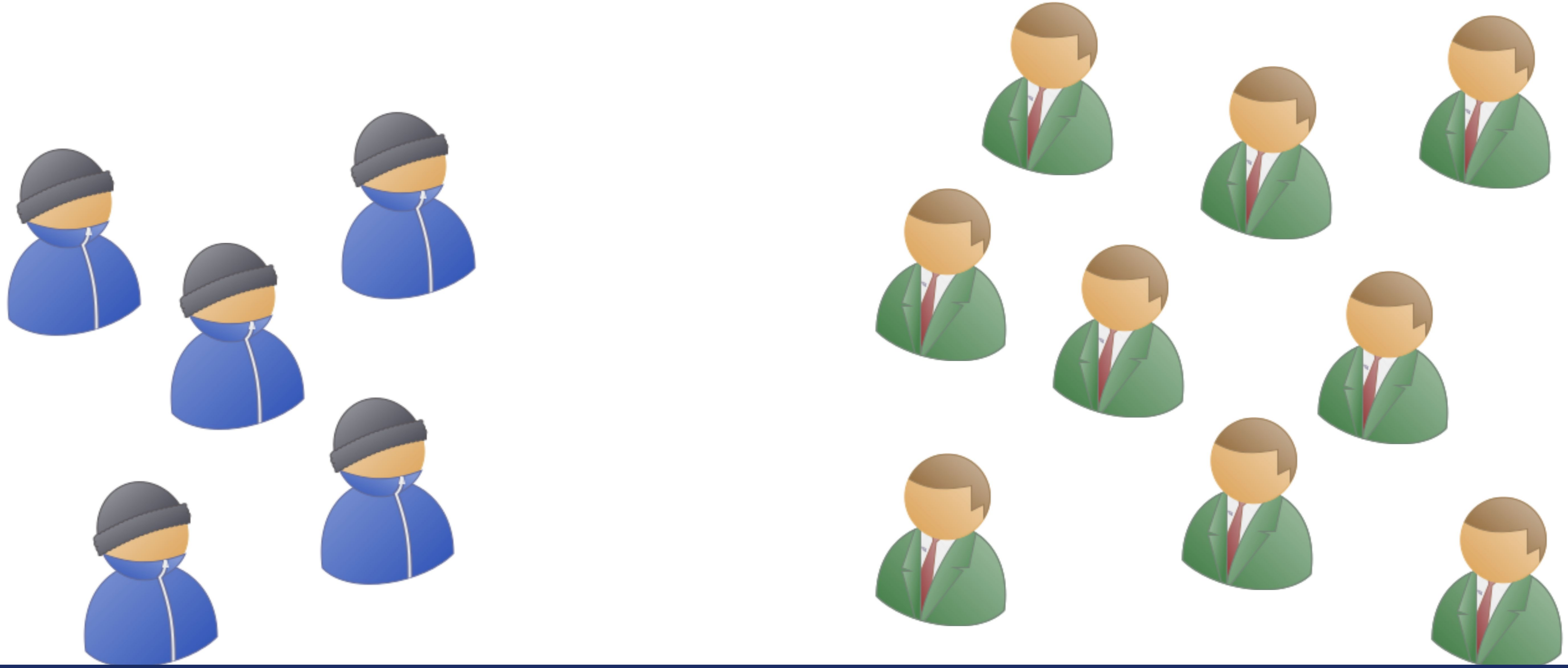
- Research questions
  - Are mandates useful?
  - When are mandates useful?

- Answer these questions using monte carlo simulations

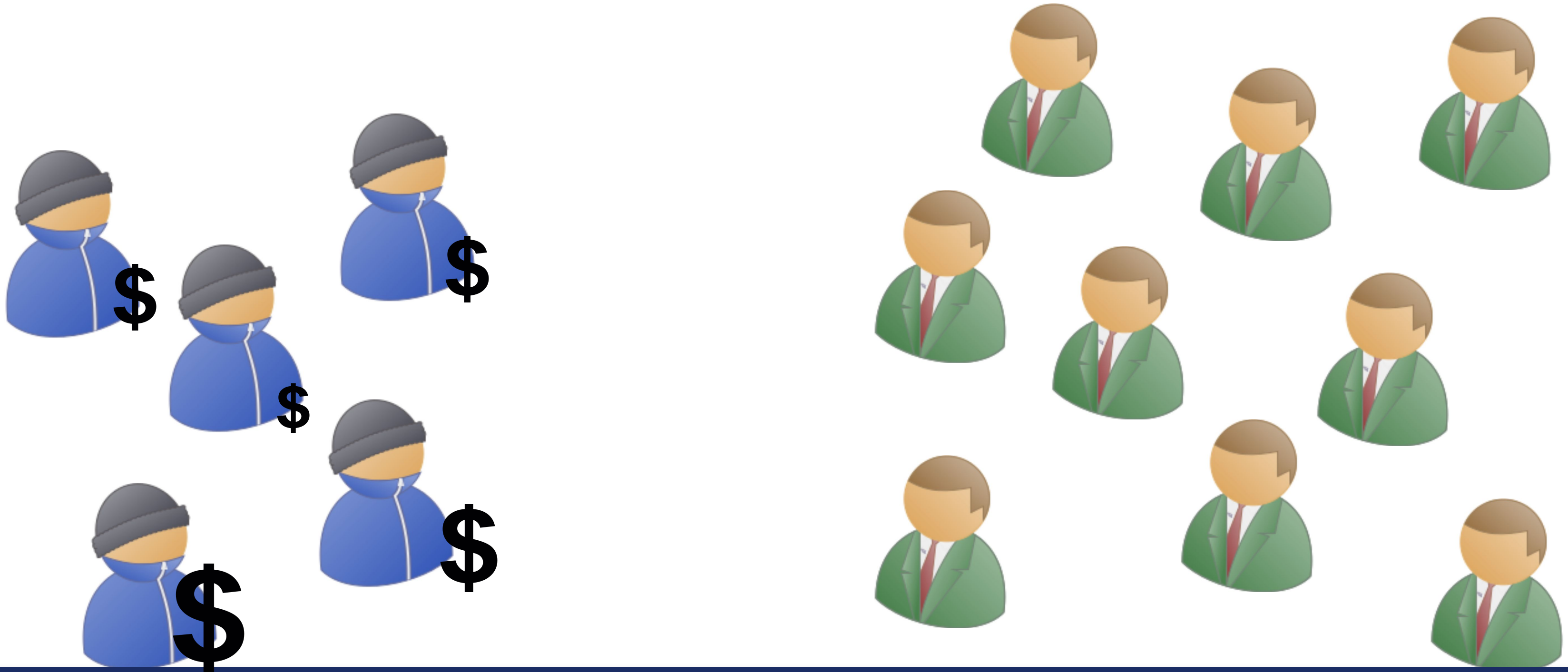
A Model Security Game



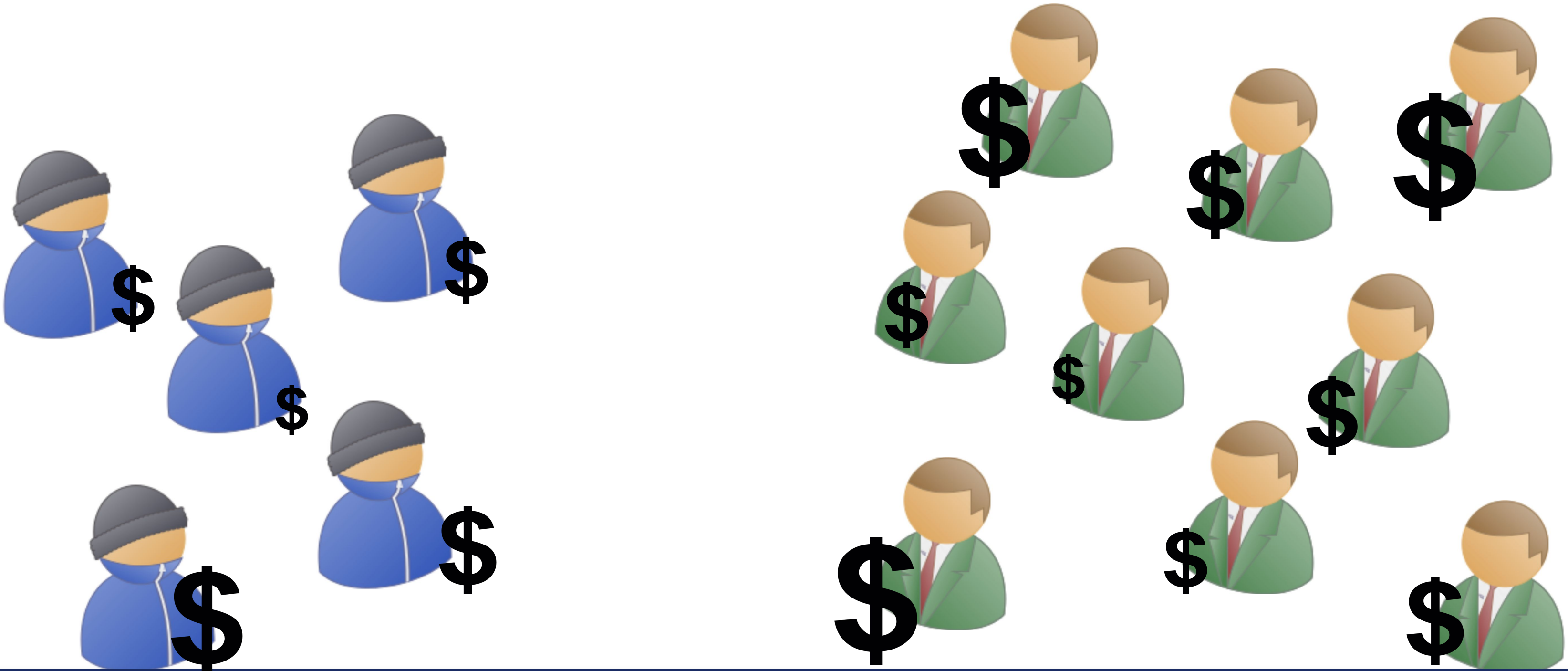
A Model Security Game



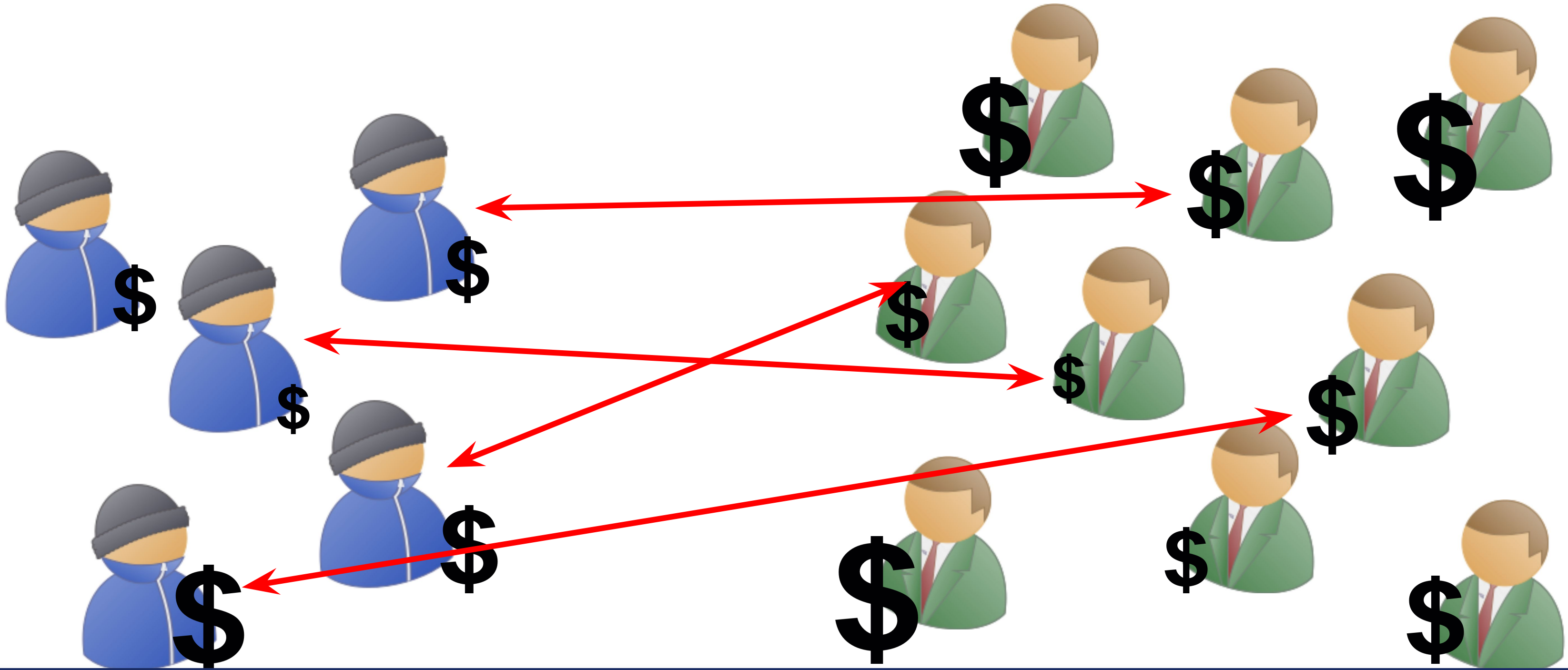
A Model Security Game



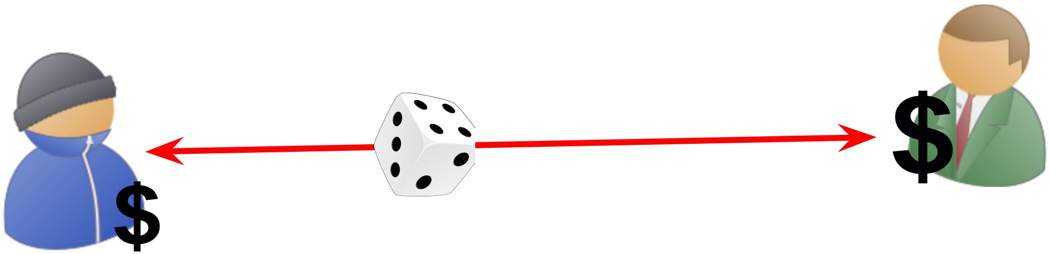
A Model Security Game



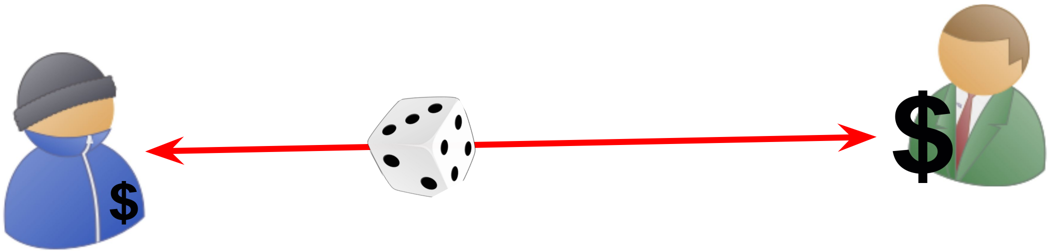
A Model Security Game



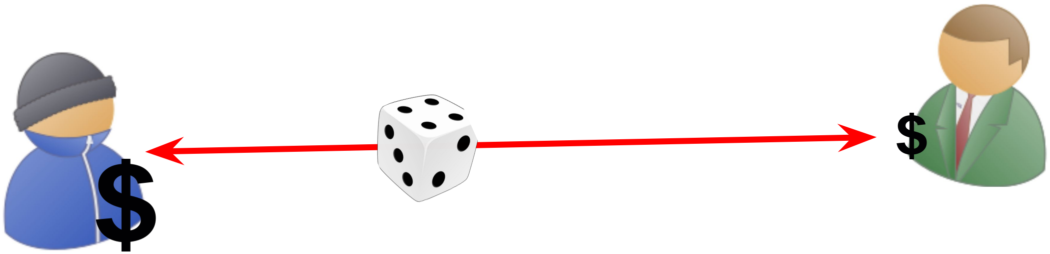
A Model Security Game



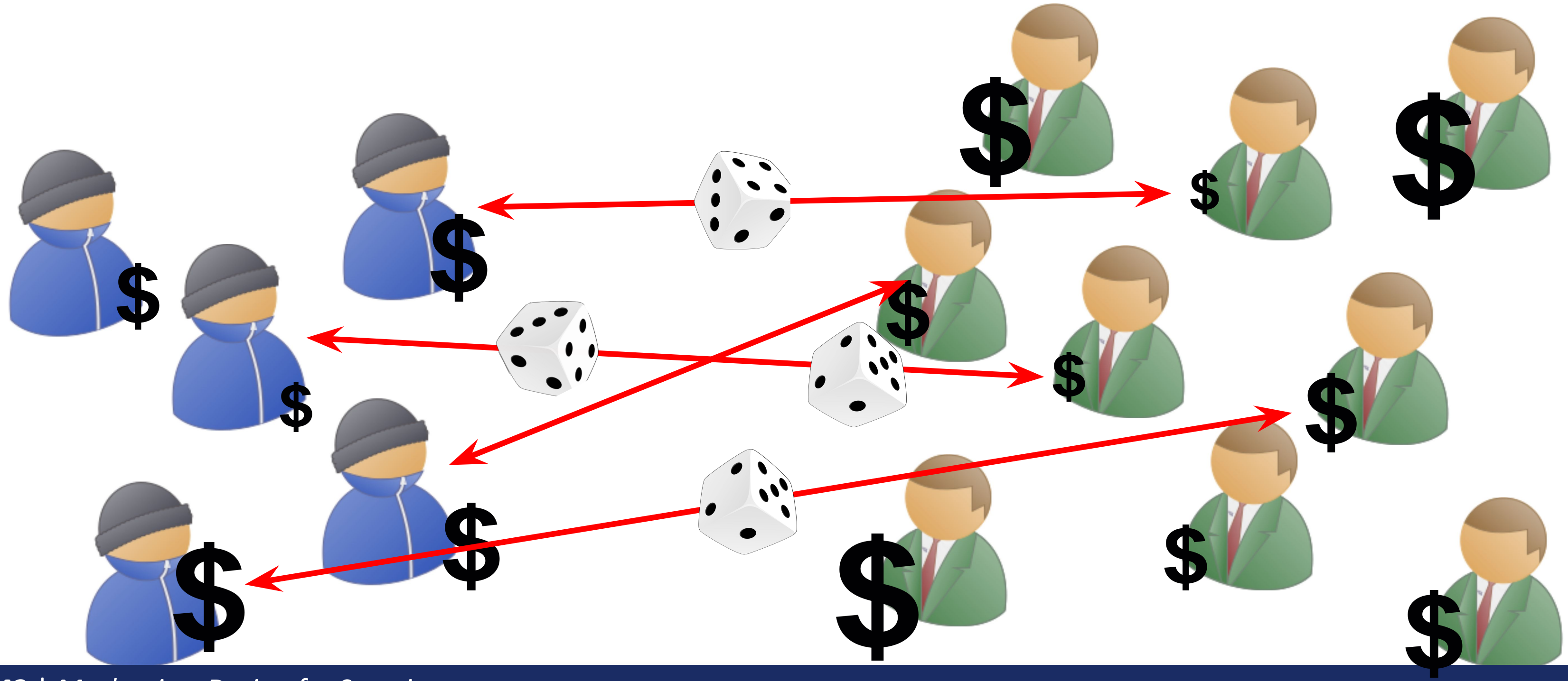
A Model Security Game



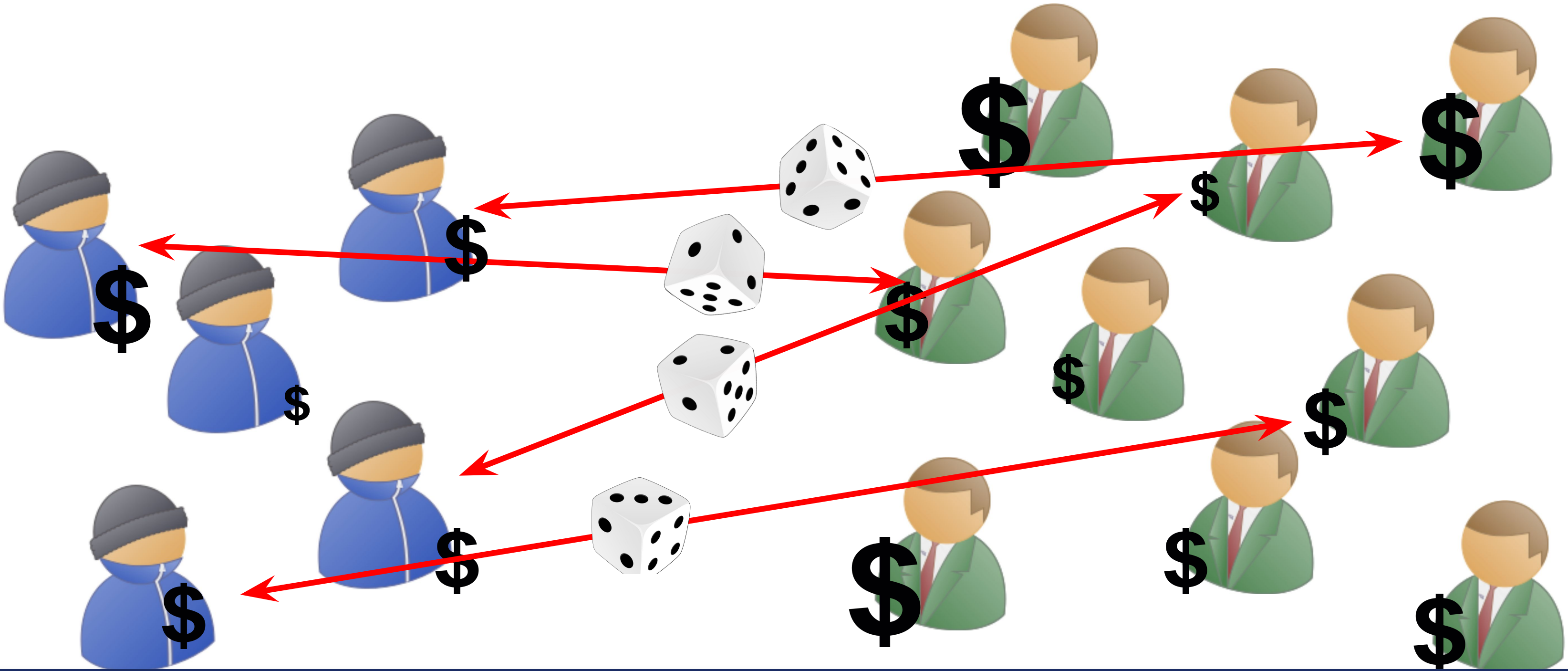
A Model Security Game



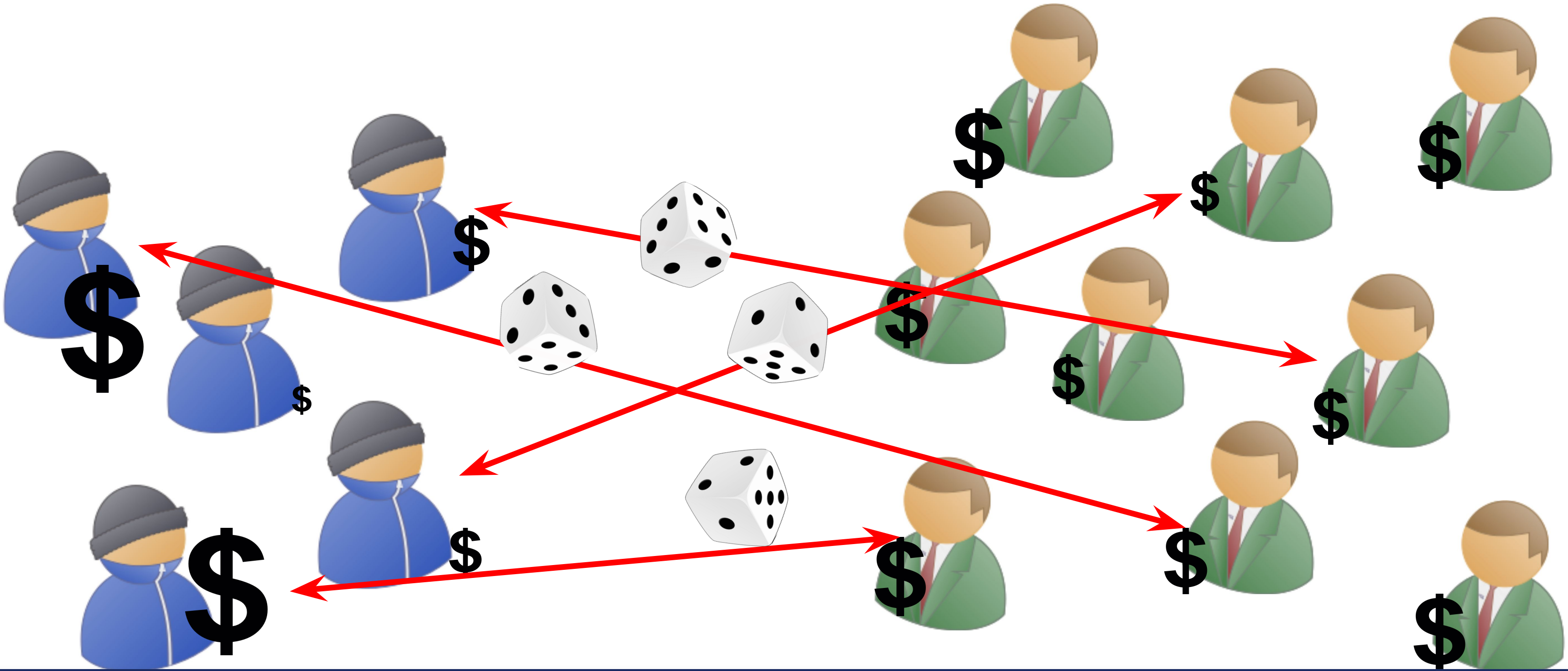
A Model Security Game



A Model Security Game



A Model Security Game



When does the game stop?

1. When all the Attackers lose all their assets

When does the game stop?

1. When all the Attackers lose all their assets

2. When all the Defenders lose all their assets

When does the game stop?

1. When all the Attackers lose all their assets

2. When all the Defenders lose all their assets

3. When the game reaches a stalemate
    a. I.e. when the collective wealth of Attackers or Defenders doesn’t change by some ε for *n* iterations

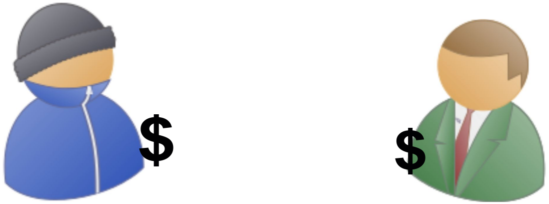
When does the game stop?

1. When all the Attackers lose all their assets

2. When all the Defenders lose all their assets

99% of games

3. When the game reaches a stalemate
    a. I.e. when the collective wealth of Attackers or Defenders doesn’t change by some ε for *n* iterations



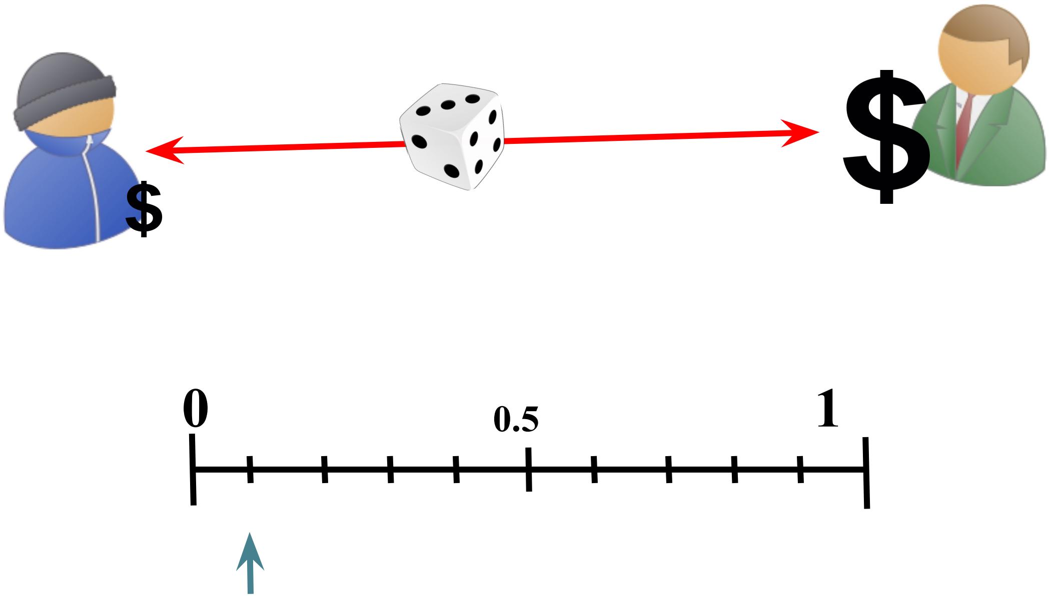
Game Parameters

Parameter Name	Description
`ATTACKERS`	Number of attackers compared to defenders, as a percentage
`INEQUALITY`	Fraction by which defender wealth distribution is scaled to create attacker wealth distribution
`ATTACK_COST`	The amount an Attacker must invest to mount an attack. Expressed as a percentage of a Defender's assets
`PAYOFF`	Max percentage of defender assets that can be taken in an attack
`MANDATE`	Percentage of defender assets that are spent on security measures
`EFFECTIVENESS`	Percentage of MANDATE by which the cost to attack a defender increases

Game Parameters

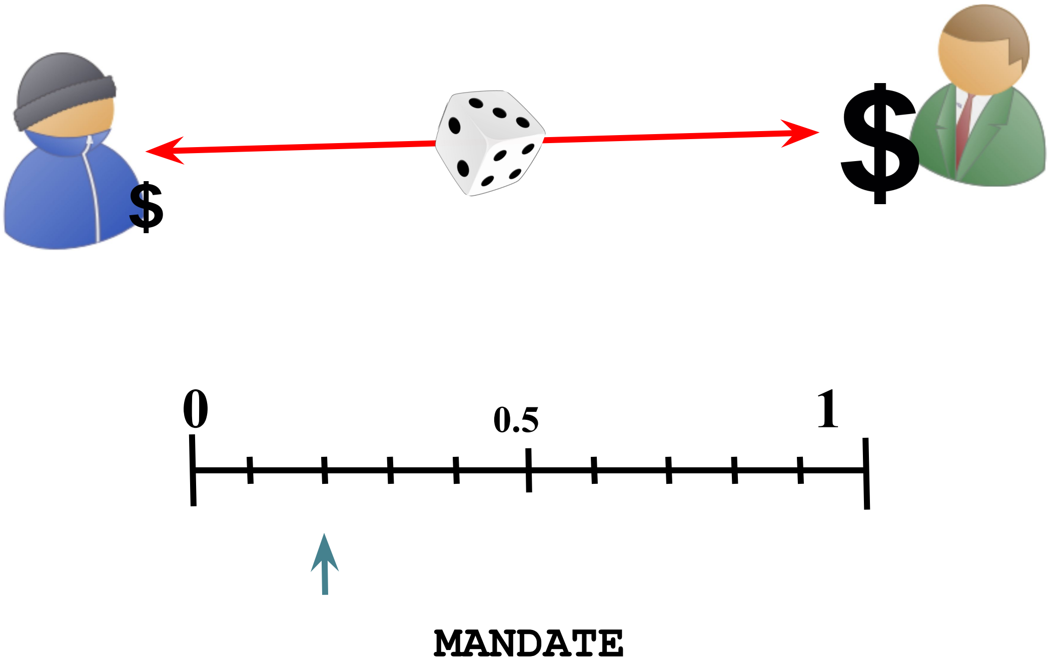
Parameter Name	Description
ATTACKERS	Number of attackers compared to defenders, as a percentage
INEQUALITY	Fraction by which defender wealth distribution is scaled to create attacker wealth distribution
ATTACK_COST	The amount an Attacker must invest to mount an attack. Expressed as a percentage of a Defender's assets
PAYOFF	Max percentage of defender assets that can be taken in an attack
MANDATE	**Percentage of defender assets that are spent on security measures**
EFFECTIVENESS	Percentage of MANDATE by which the cost to attack a defender increases

MANDATE parameter

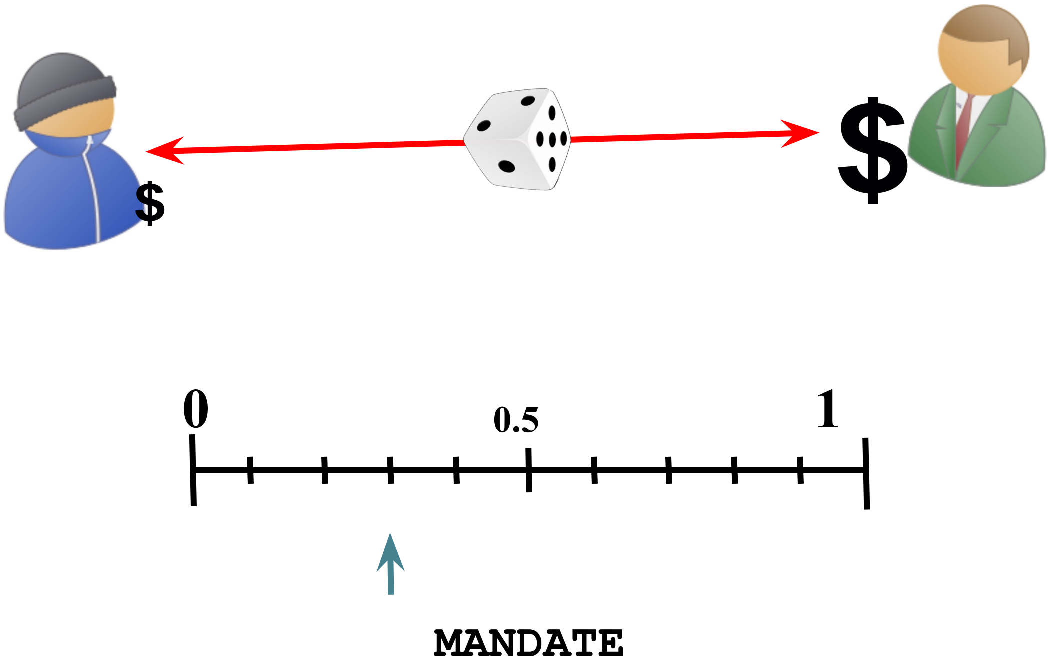


MANDATE

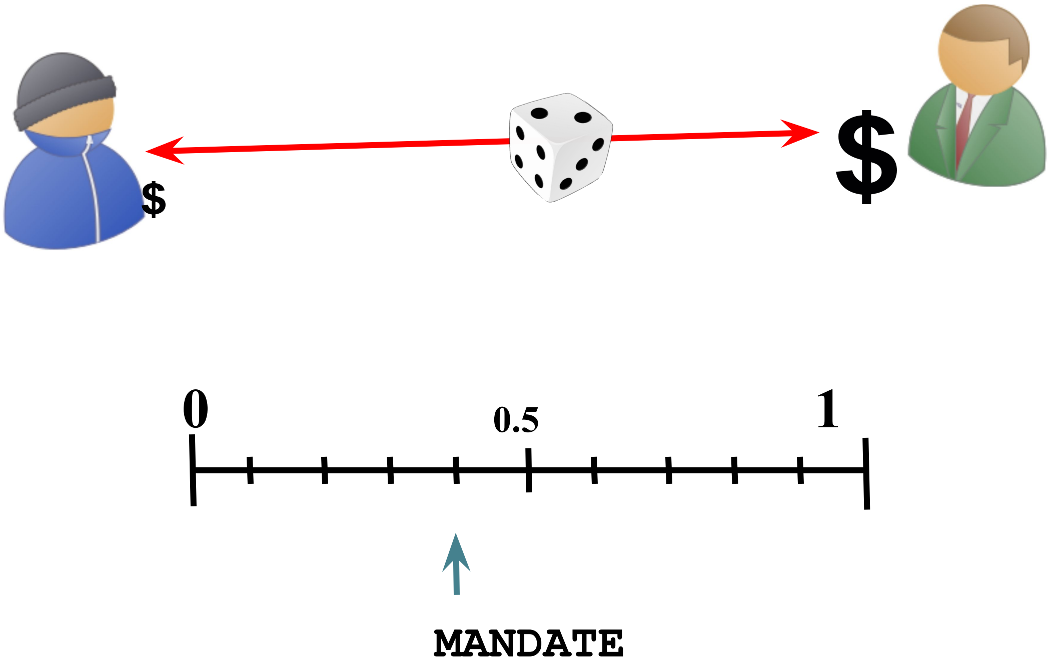
MANDATE parameter



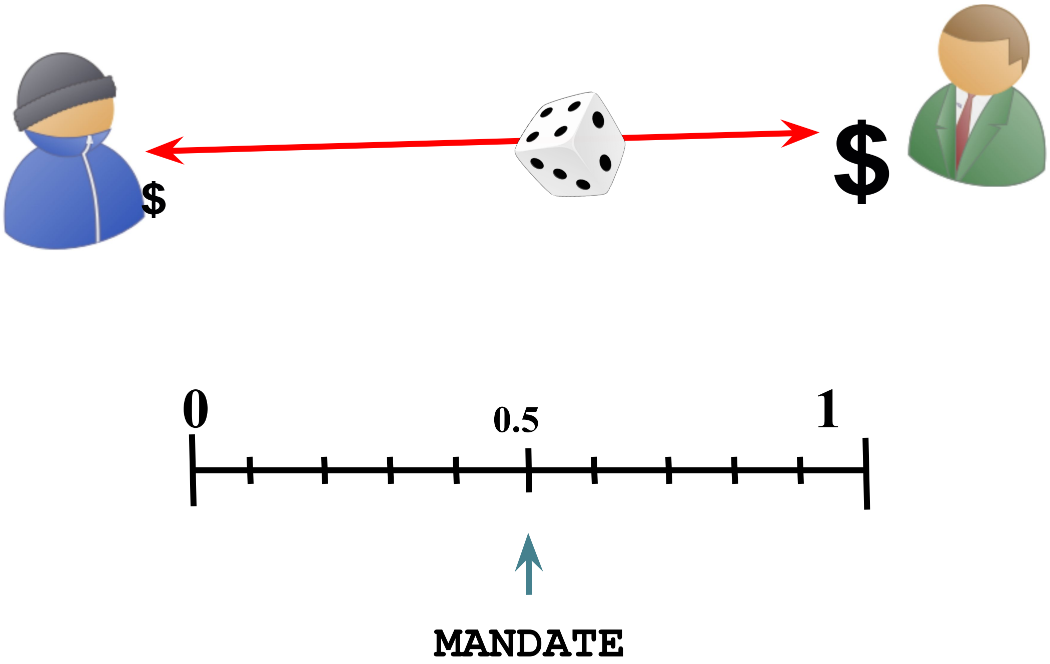
MANDATE parameter



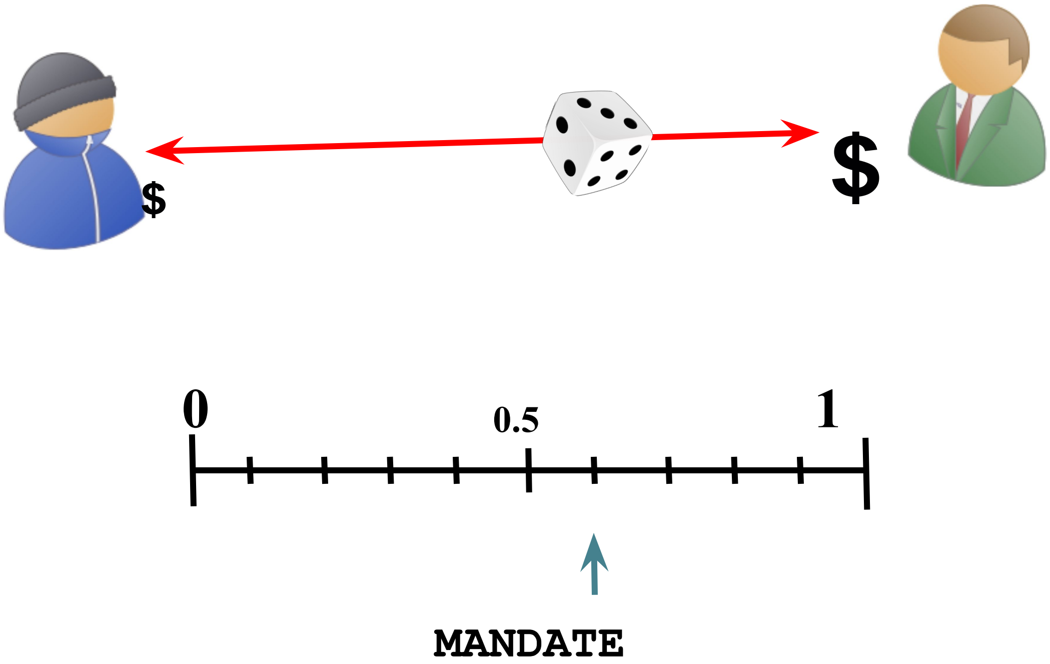
MANDATE parameter



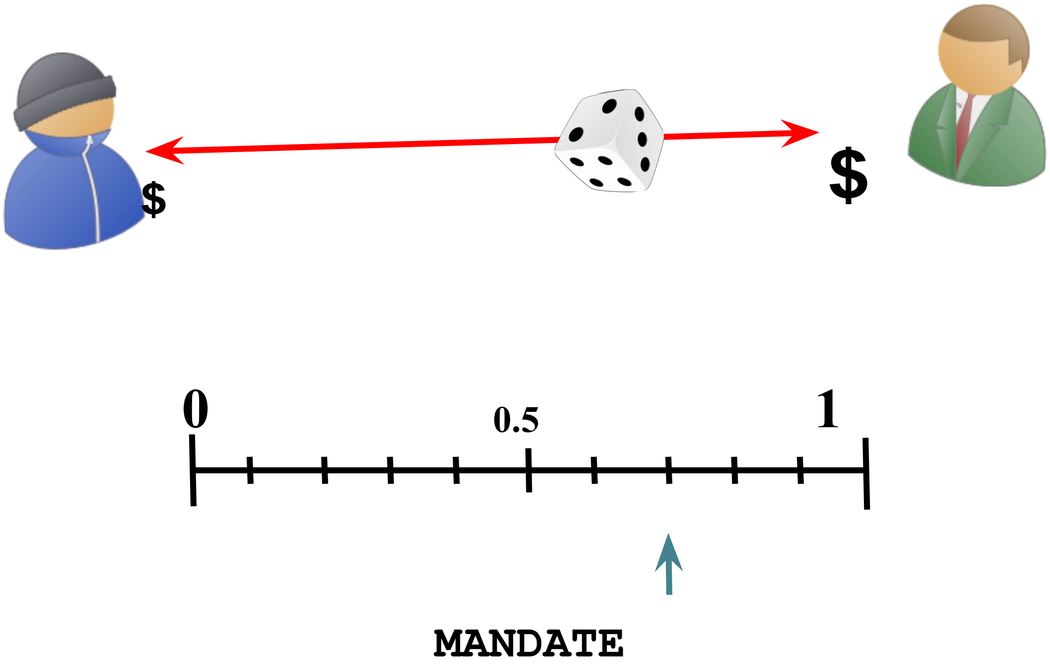
MANDATE parameter



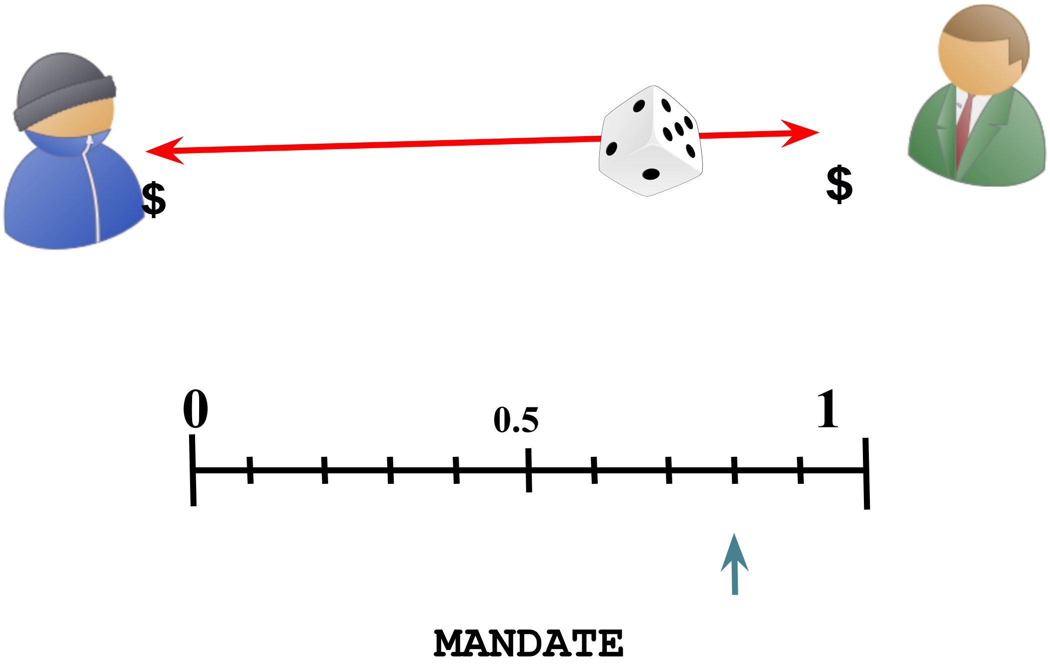
MANDATE parameter



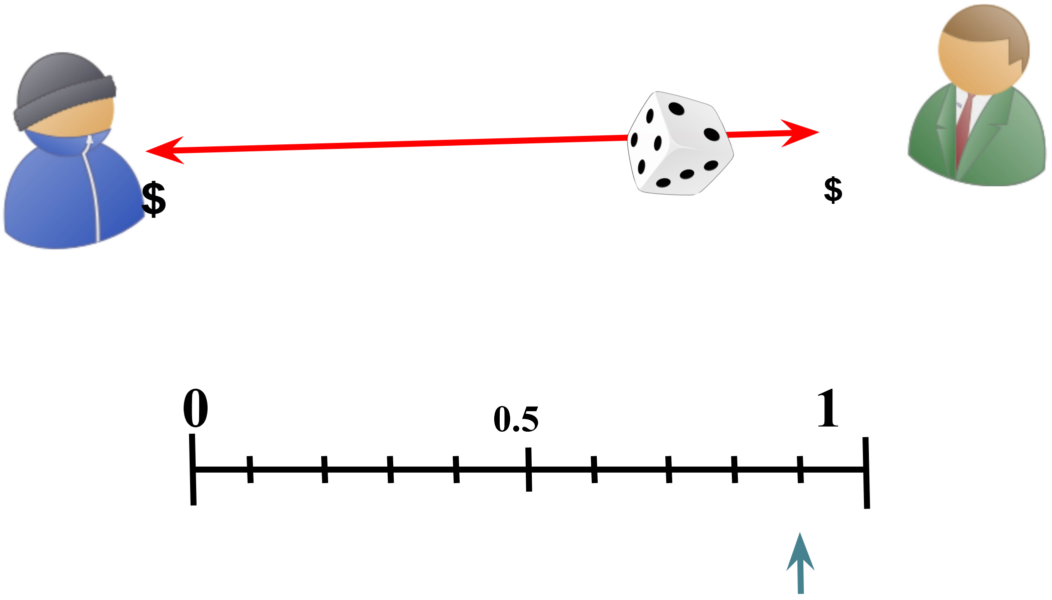
`MANDATE` parameter



MANDATE parameter



MANDATE parameter



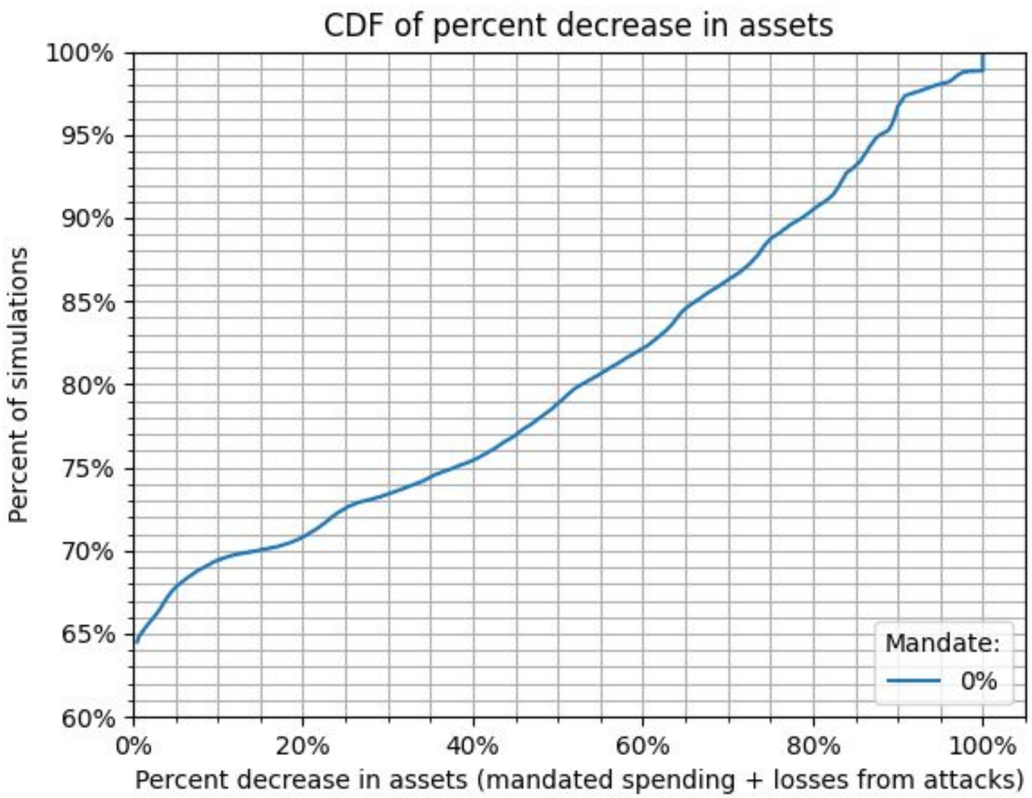
MANDATE

Game Parameters

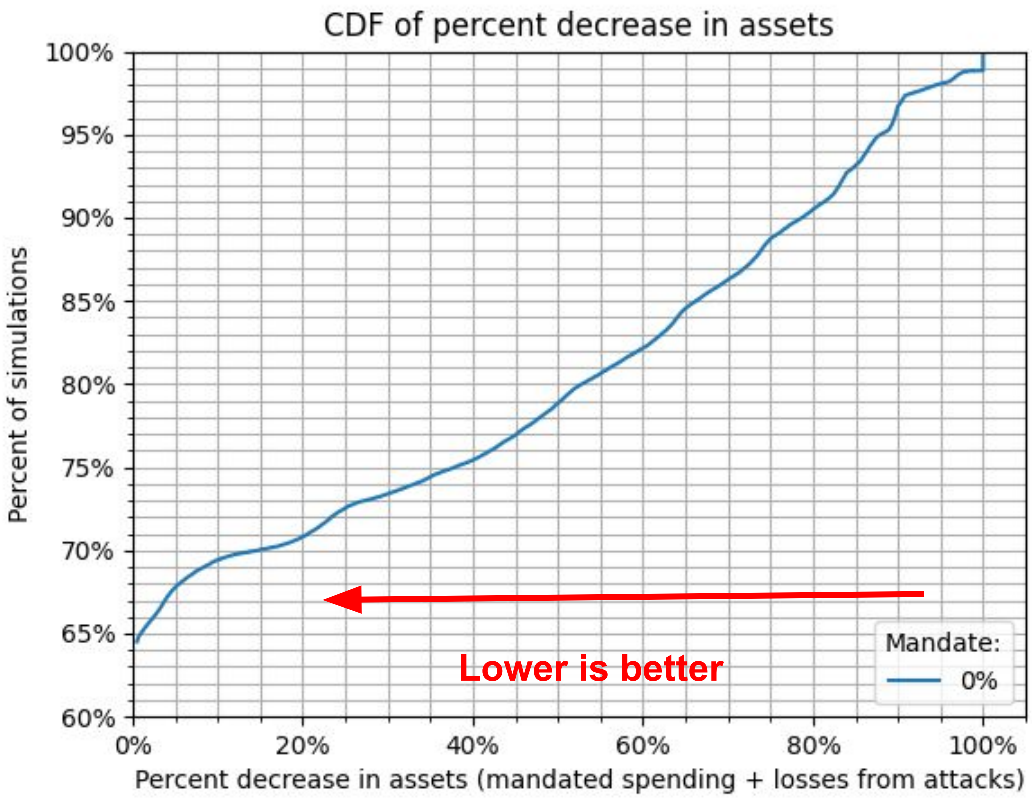
Parameter Name	Description
ATTACKERS	Number of attackers compared to defenders, as a percentage
INEQUALITY	Fraction by which defender wealth distribution is scaled to create attacker wealth distribution
ATTACK_COST	The amount an Attacker must invest to mount an attack. Expressed as a percentage of a Defender's assets
PAYOFF	Max percentage of defender assets that can be taken in an attack
MANDATE	Percentage of defender assets that are spent on security measures
EFFECTIVENESS	Percentage of MANDATE by which the cost to attack a defender increases

10^6 possible game configurations

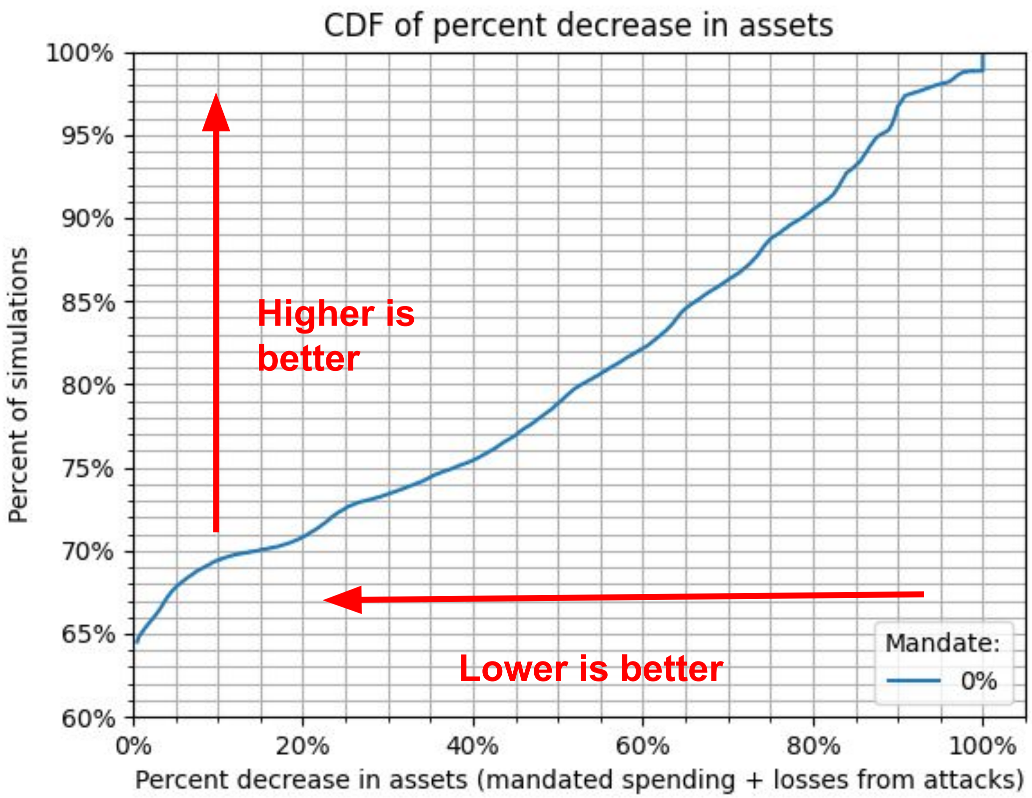
How does MANDATE parameter affect game outcomes?



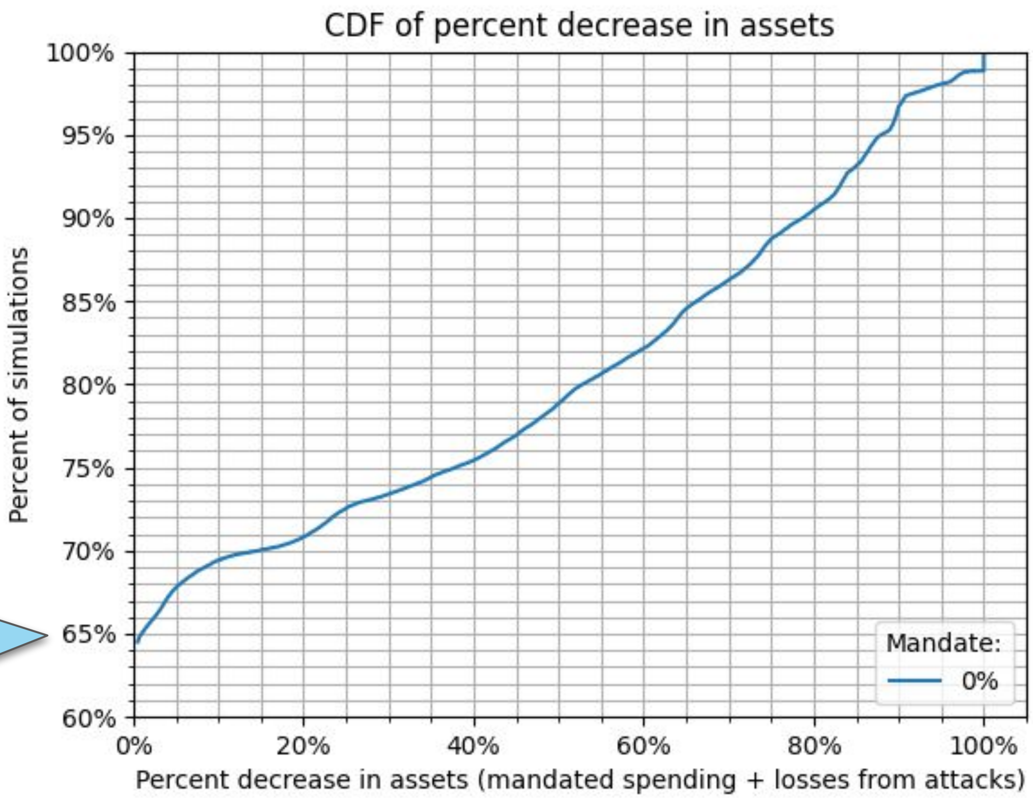
How does MANDATE parameter affect game outcomes?



How does MANDATE parameter affect game outcomes?

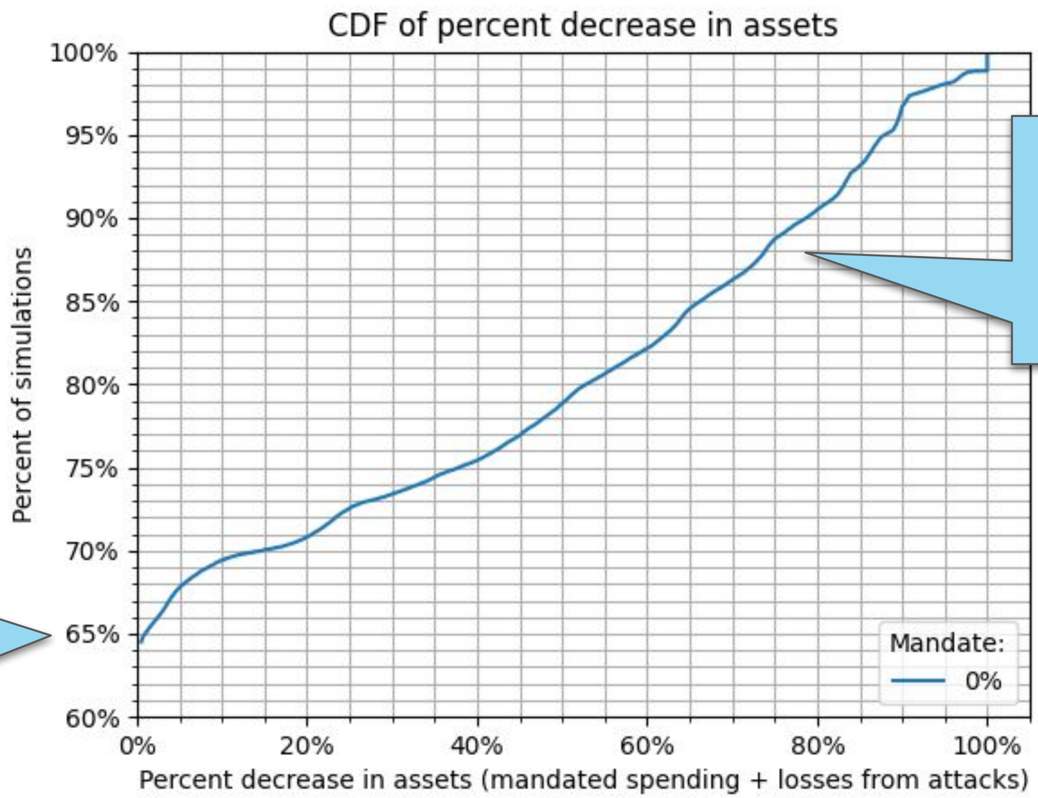


How does MANDATE parameter affect game outcomes?



Mandates provides no benefit for 65% of games

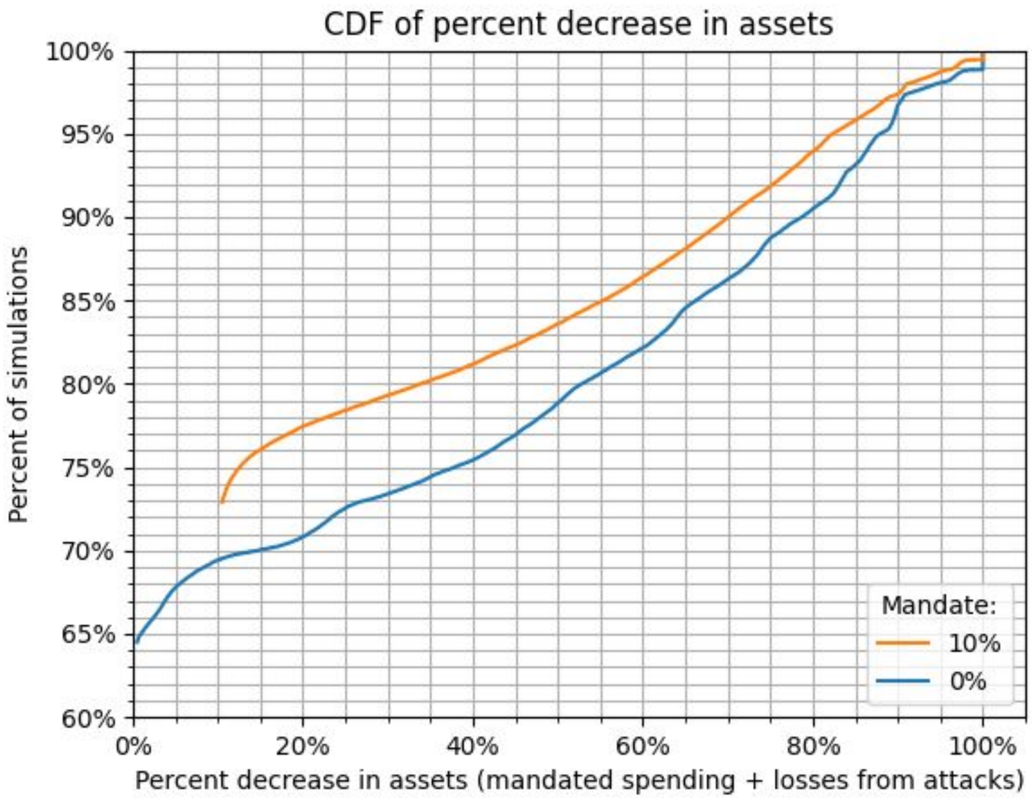
How does MANDATE parameter affect game outcomes?



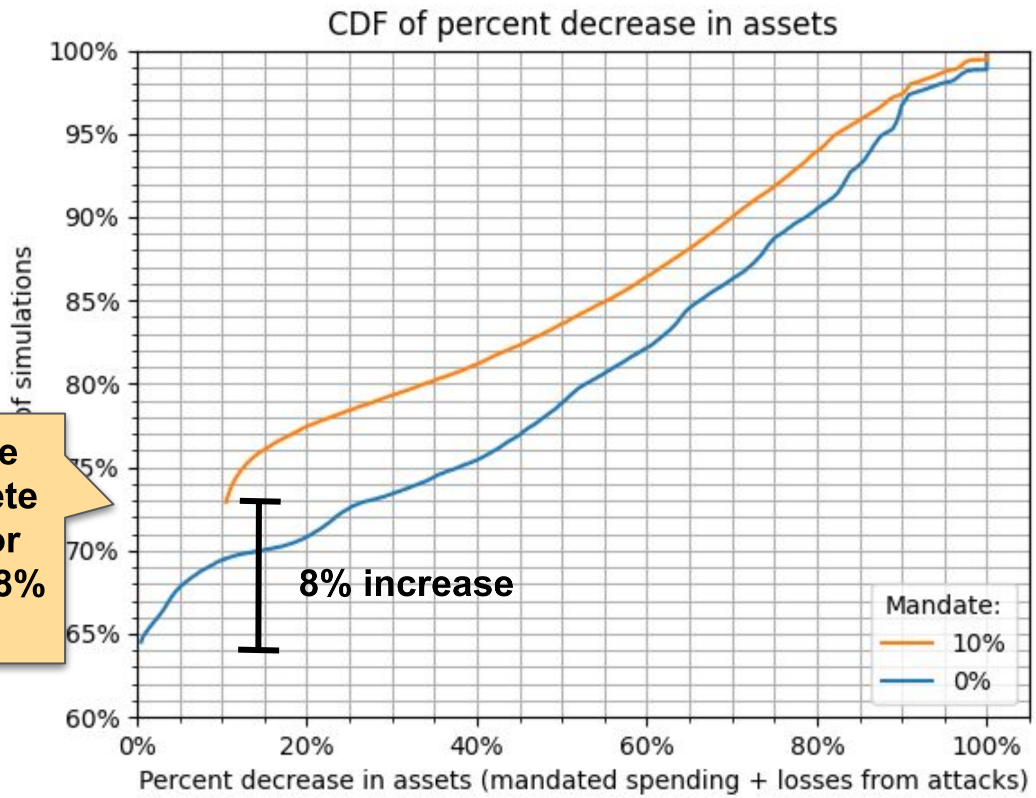
Mandates provides no benefit for 65% of games

Mandates could be beneficial for remaining 35% of games

How does MANDATE parameter affect game outcomes?

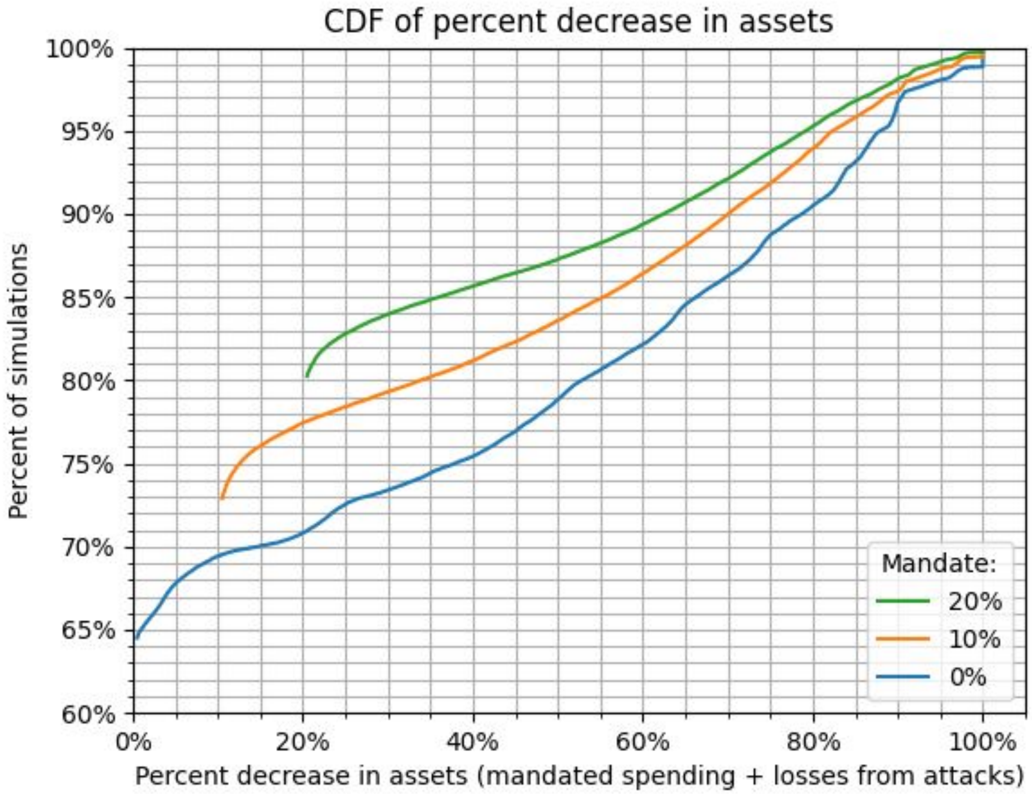


How does MANDATE parameter affect game outcomes?

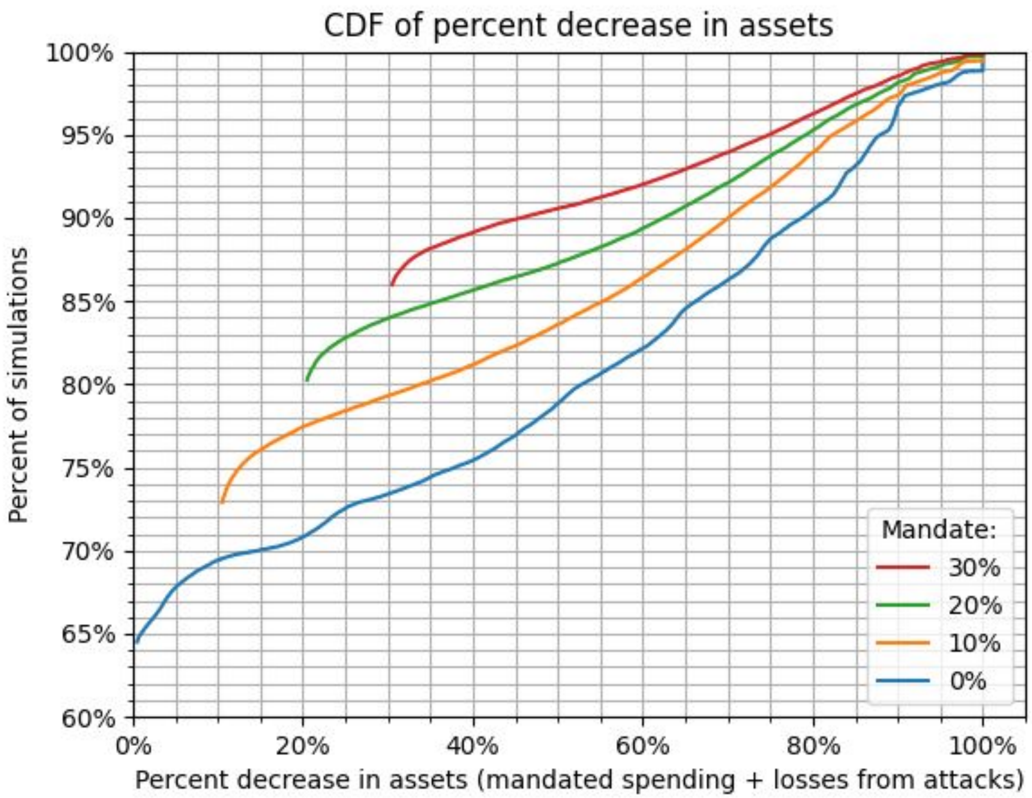


10% mandate offers complete protection for an additional 8% of games

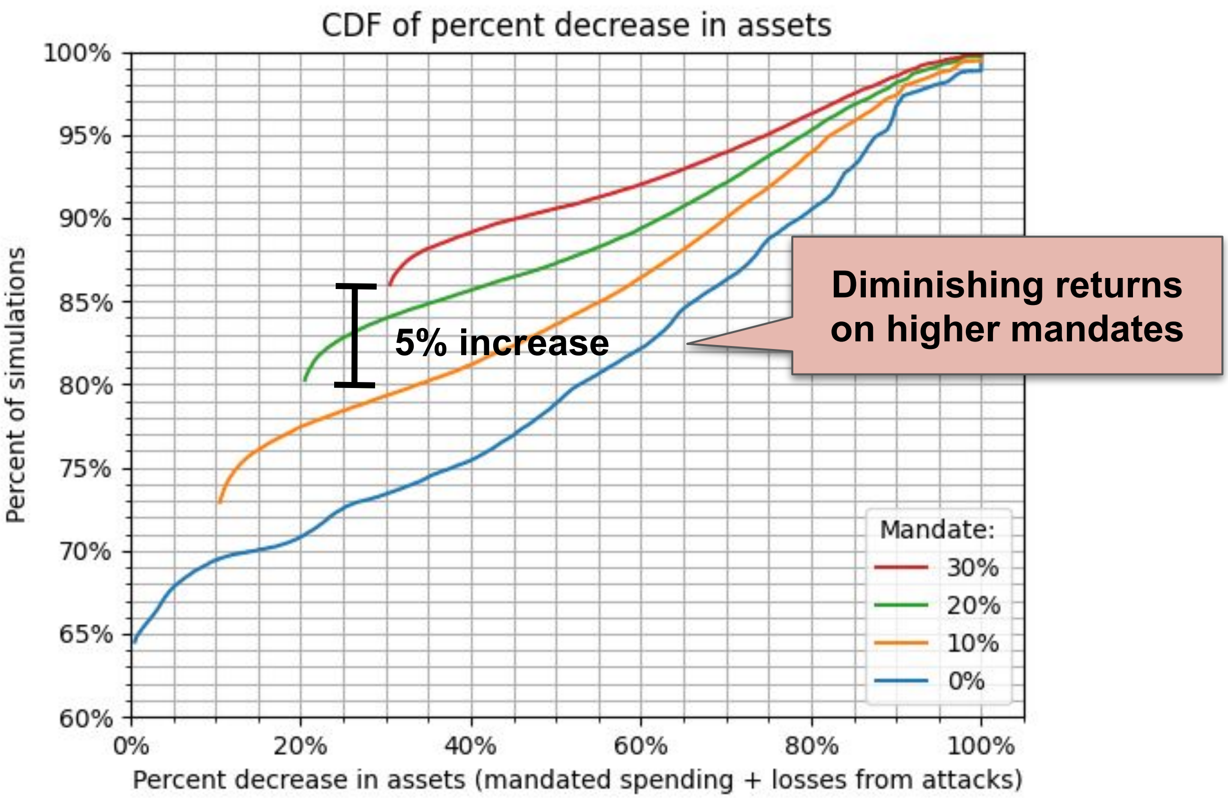
How does MANDATE parameter affect game outcomes?



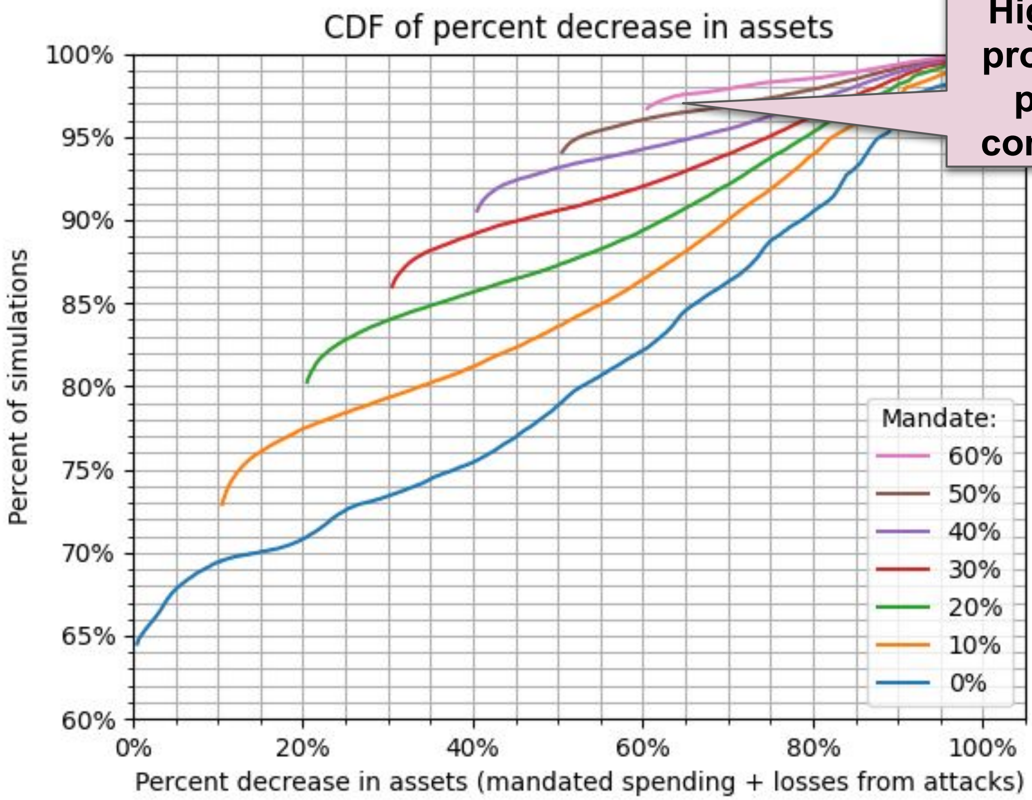
How does MANDATE parameter affect game outcomes?



How does MANDATE parameter affect game outcomes?



How does MANDATE parameter affect game outcomes?

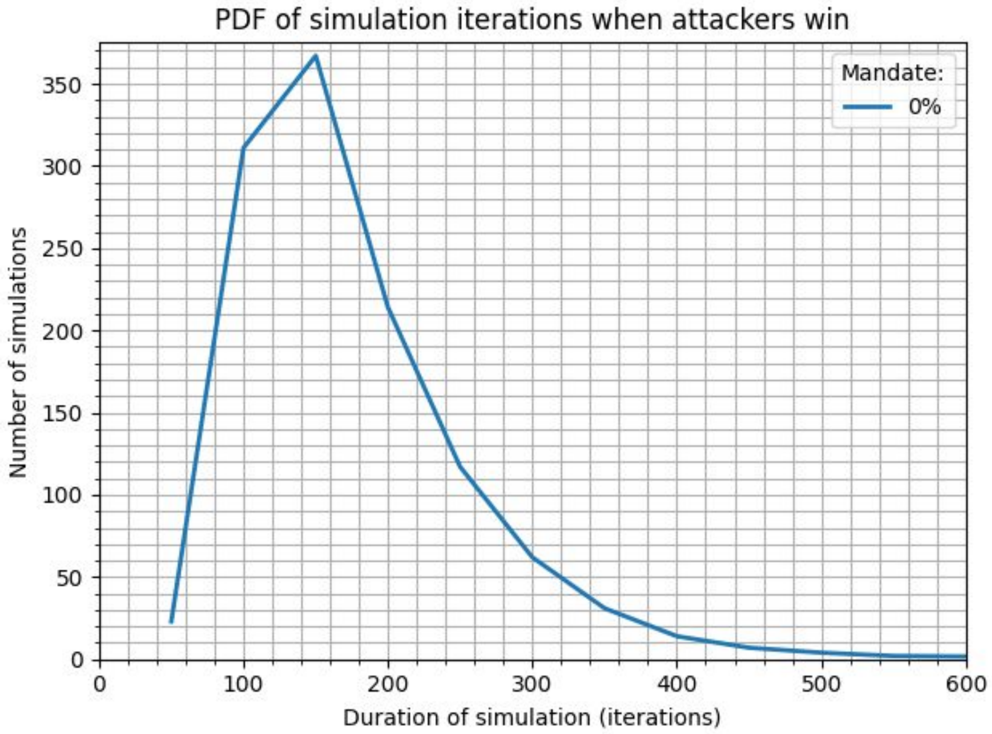


Higher mandates provide near-total protection but come at high cost

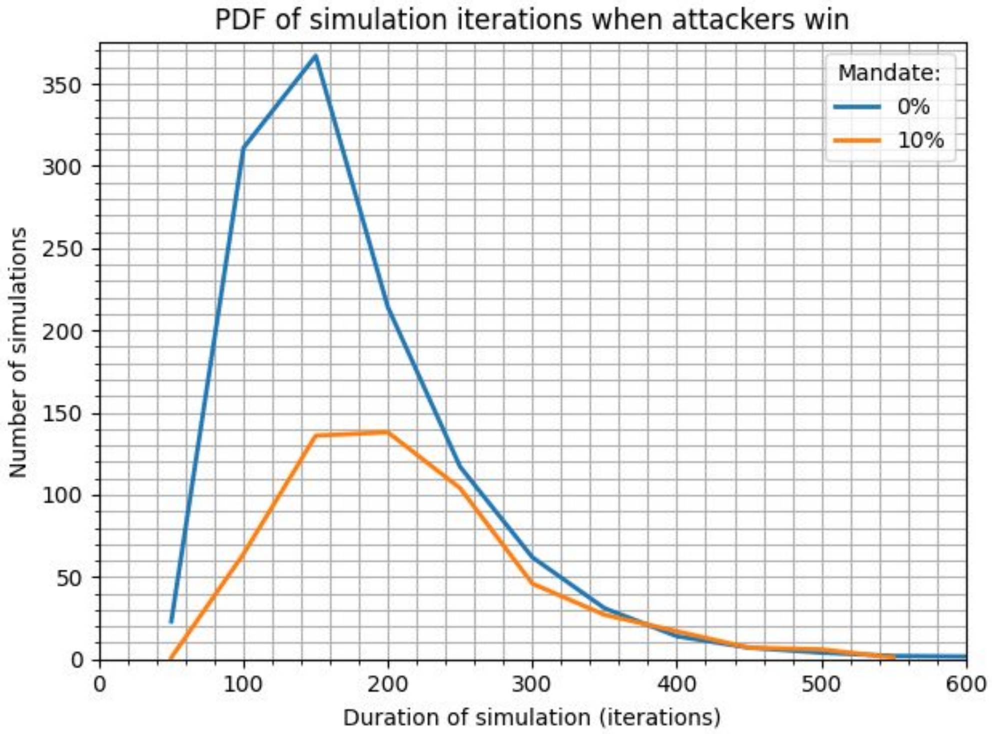
Takeaways

Under reasonable assumptions, mandates can improve overall outcomes for defenders (up to a point)

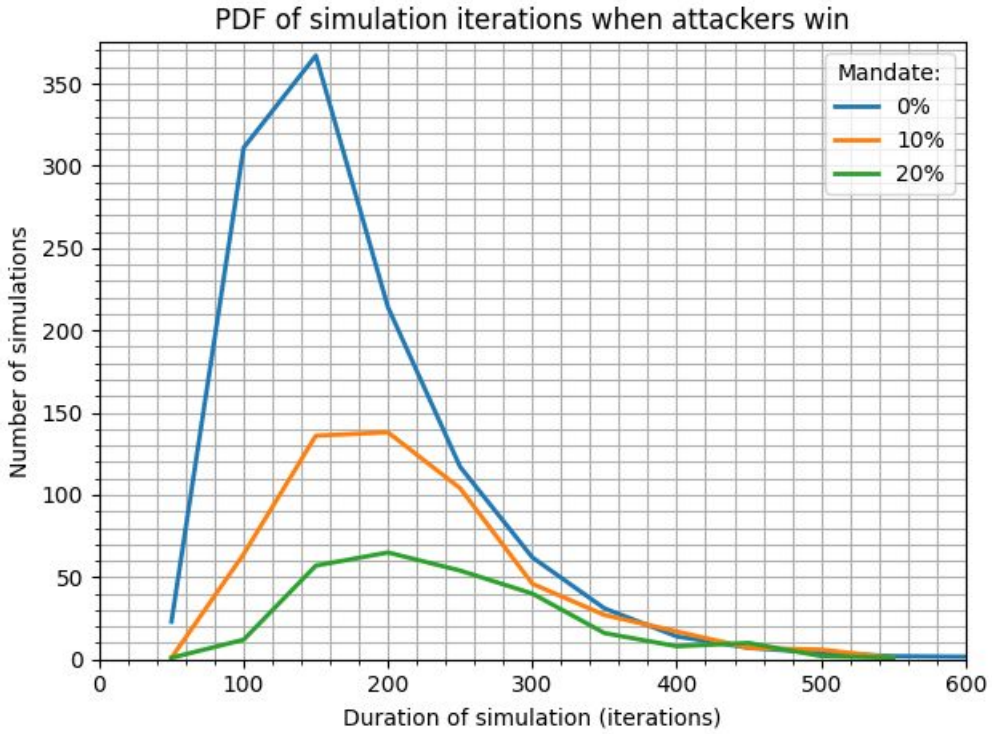
When attackers win...



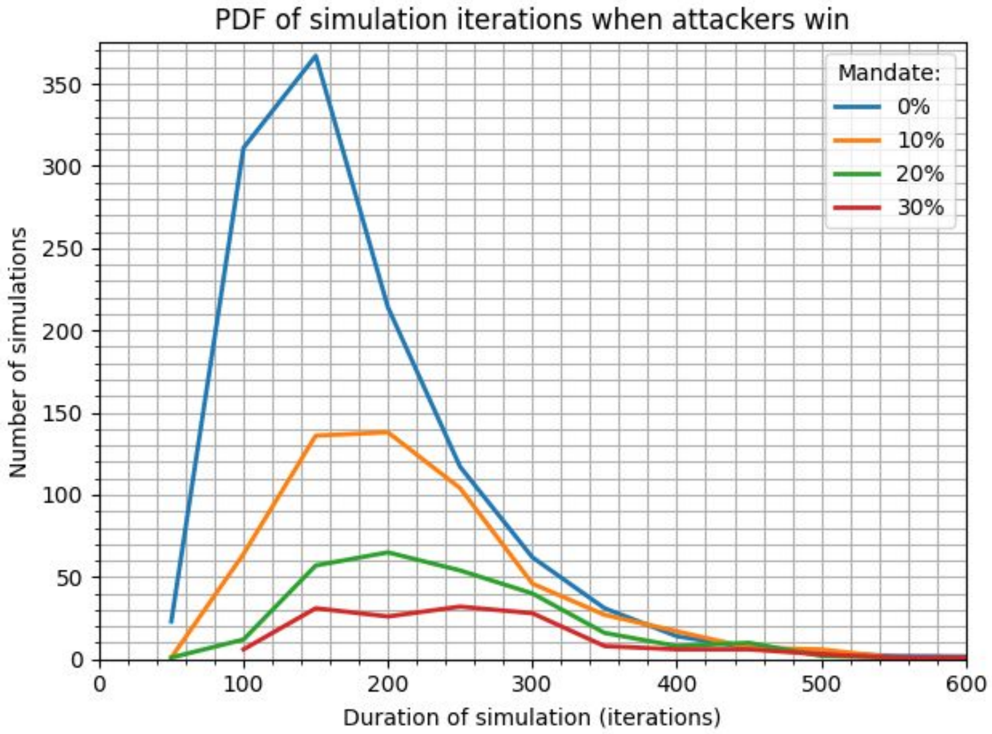
When attackers win…



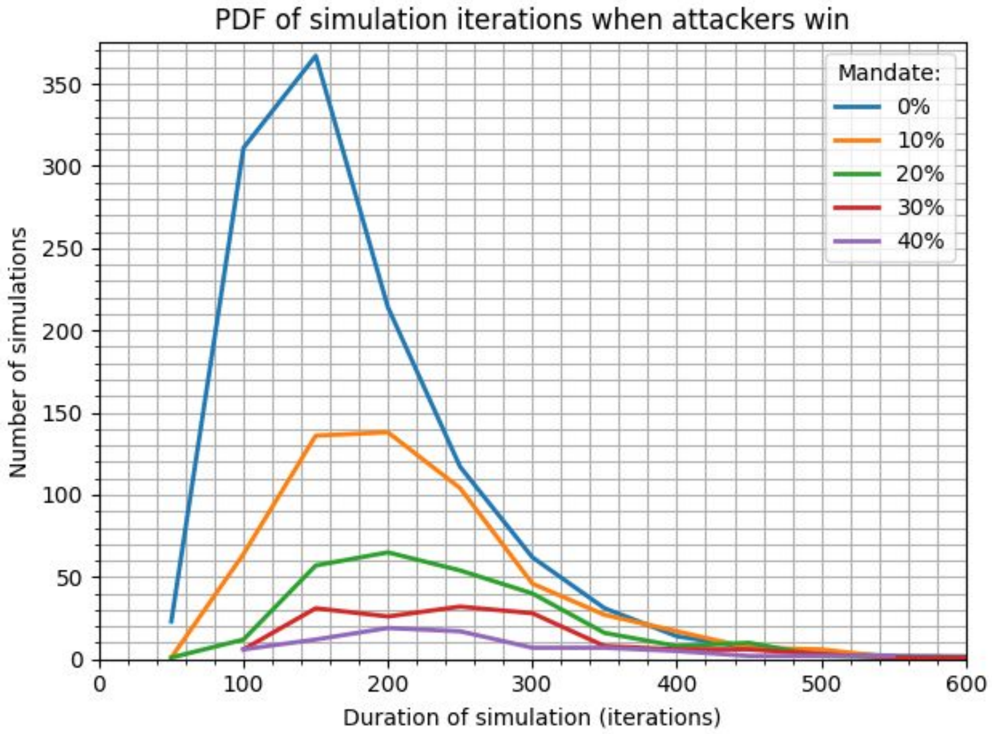
When attackers win...



When attackers win…



When attackers win…

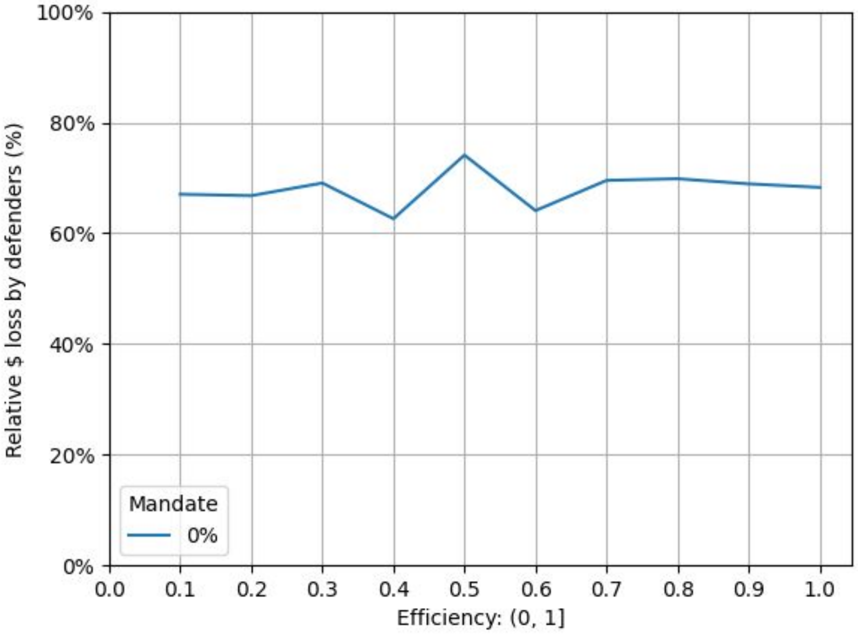


Takeaways

Even when Attackers win, mandates slow Attacker progress

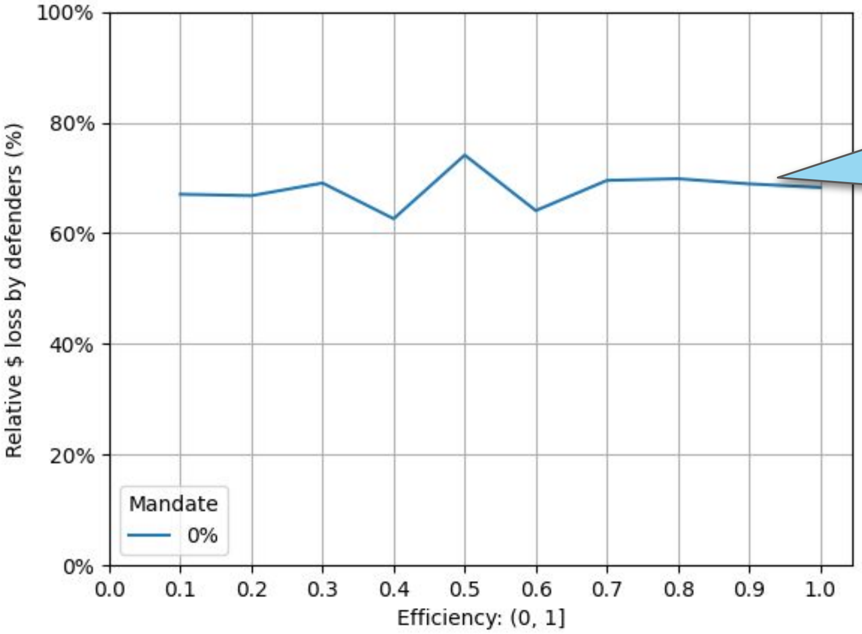
Efficiency vs. Losses

Lower is better



Efficiency vs. Losses

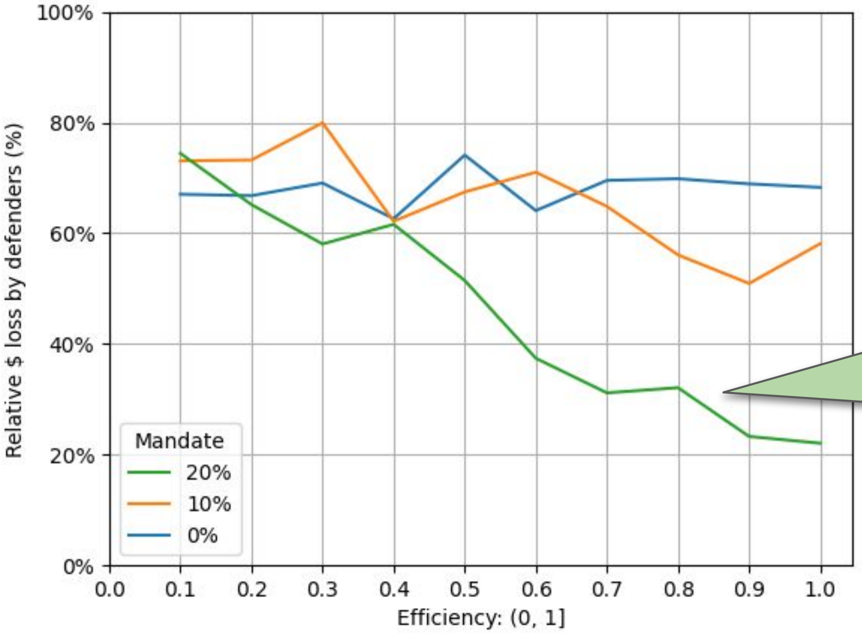
Lower is better



Losses are independent of efficiency when no investments made

Efficiency vs. Losses

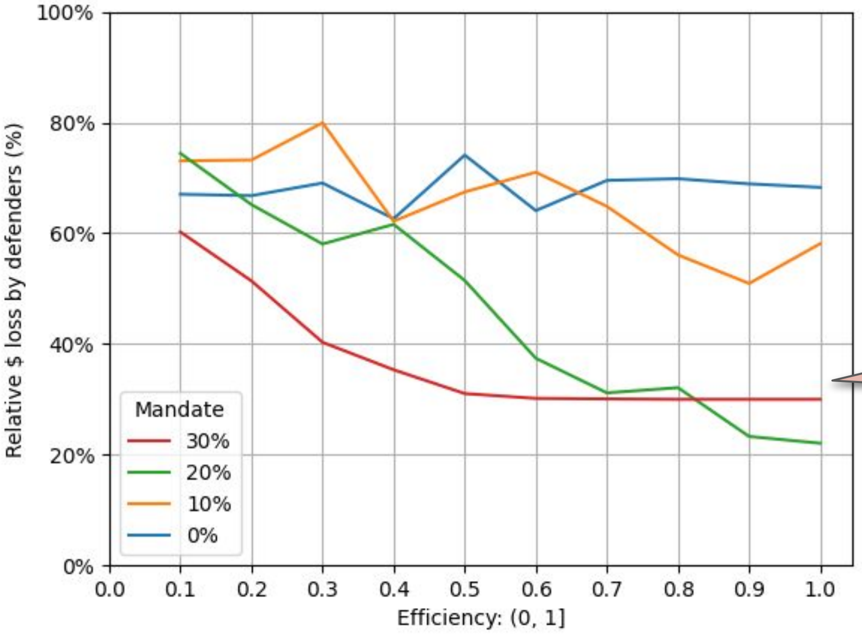
Lower is better



Losses are inversely proportional with efficiency at a 20% mandate

Efficiency vs. Losses

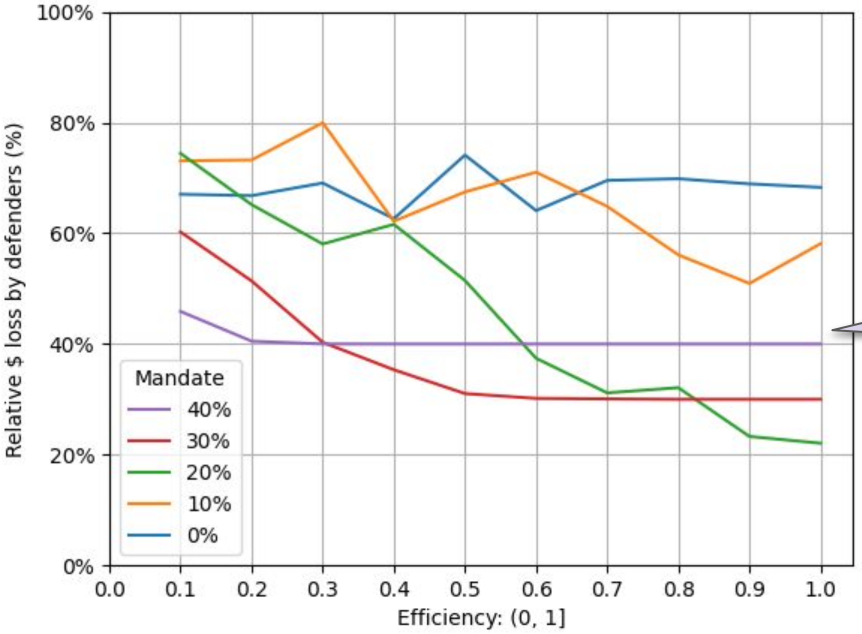
Lower is better



This trendline starts to flatten at a 30% mandate

Efficiency vs. Losses

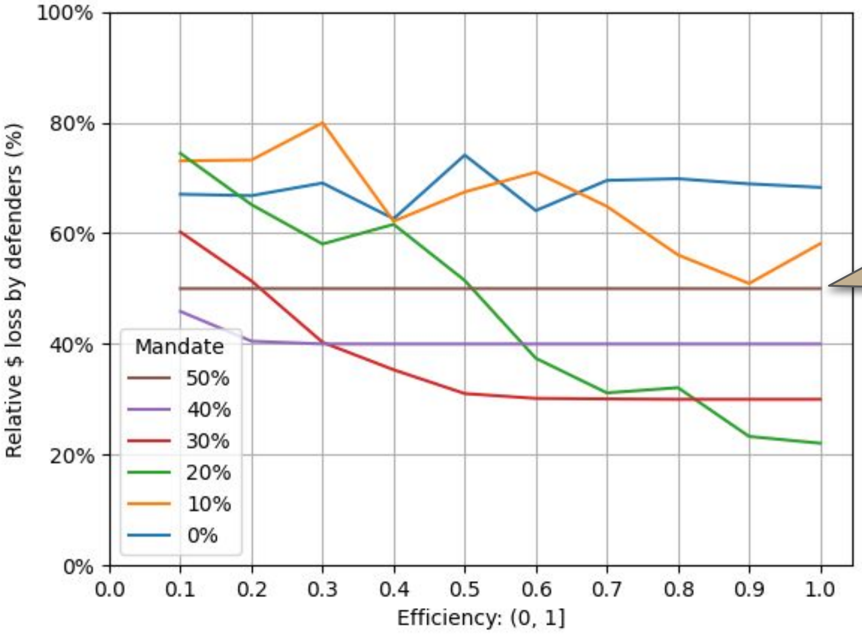
Lower is better



A 40% mandate is only useful when Efficiency is low

Efficiency vs. Losses

Lower is better

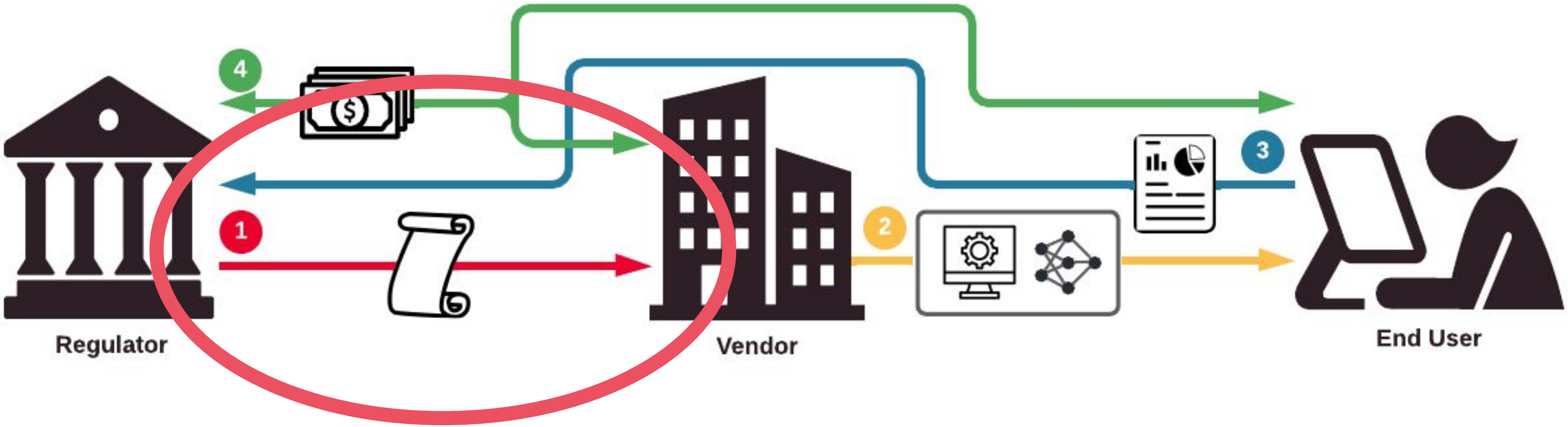


A 50% mandate is suboptimal for all values of `Efficiency`

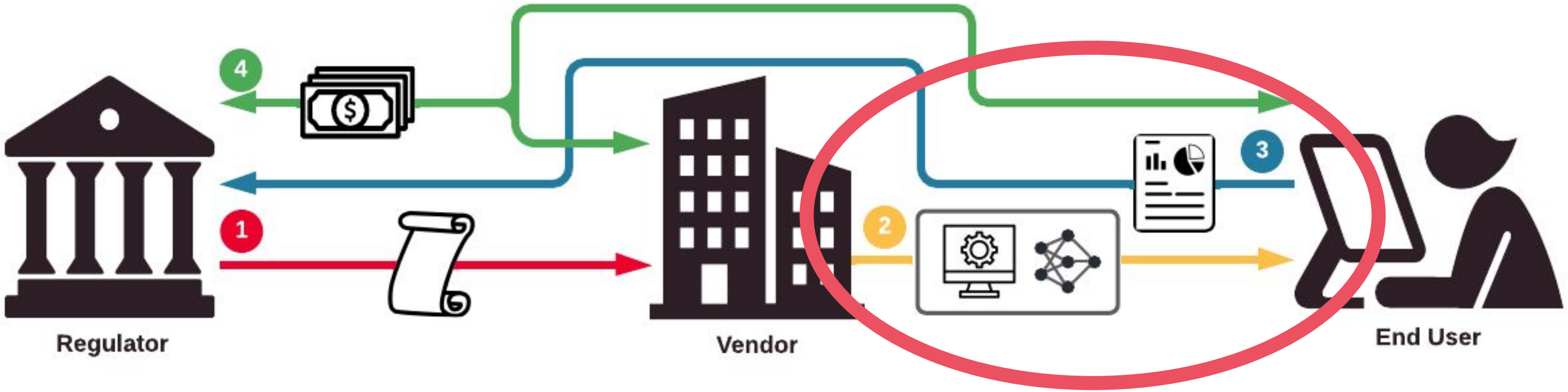
Takeaways

Certain mandate levels incentivize more efficient security solutions

COMMAND Overview



COMMAND Overview



How do we measure and report security overheads?

- Problem: Overheads are individualized
  - User, Workload-, and System-dependent
  - We need on-device, in-situ measurements

How do we measure and report security overheads?

- Problem: Overheads are individualized
  - User, Workload-, and System-dependent
  - We need on-device, in-situ measurements

- Solution: Train a model that predicts performance overhead due to security
  - Data captured from hardware performance counters (available widely)
  - Tiny DNN-based model (4 layers, 12 KB total → could be implemented in HW)
  - Training data: compare program runs with and without security defense

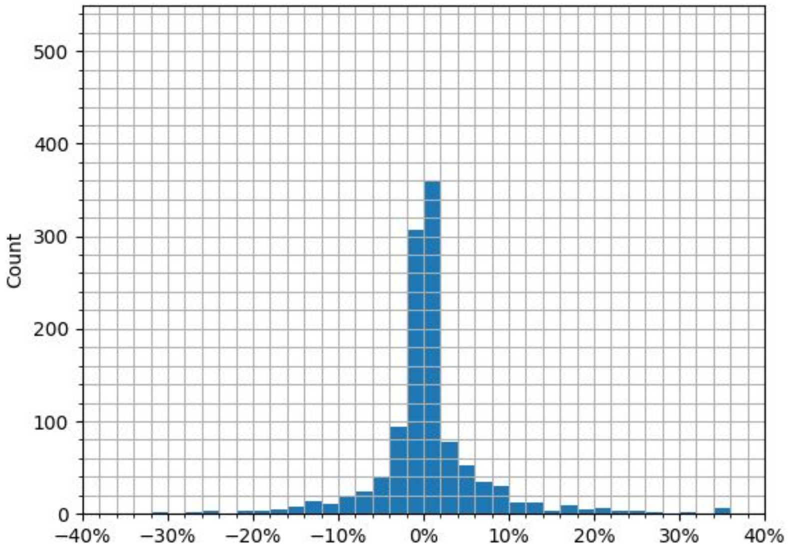
How do we measure and report security overheads?

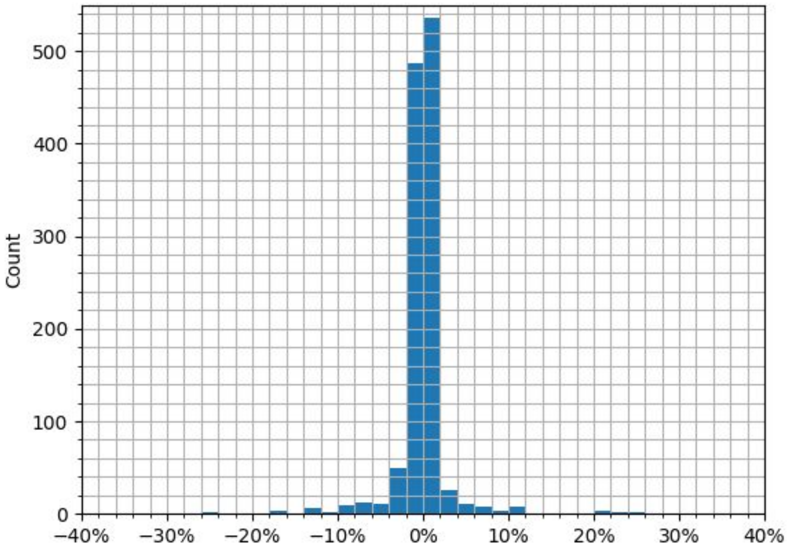
- Problem: Overheads are individualized
  - User, Workload-, and System-dependent
  - We need on-device, in-situ measurements
- Solution: Train a model that predicts performance overhead due to security
  - Data captured from hardware performance counters (available widely)
  - Tiny DNN-based model (4 layers, 12 KB total → could be implemented in HW)
  - Training data: compare program runs with and without security defense
- User collects data and submits it for rebates
  - On-device, longitudinal aggregation prevents privacy loss
  - Asymmetric crypto is used to prevent forgery

Use case: memory safety

- 39 of the 58 0-day attacks last year were due to lack of memory safety
- Like three use cases discussed before, many ways to solve
- But nudge is needed to get solutions in the market
- We model one recent memory safety solution (NoFAT, ISCA 2021)
    - The memory safety checks are entangled with regular source instructions

Results (Hardware Support for Software Security)





Absolute error: *(predicted - actual)*

Relative error: *(actual/predictions)* − *1*

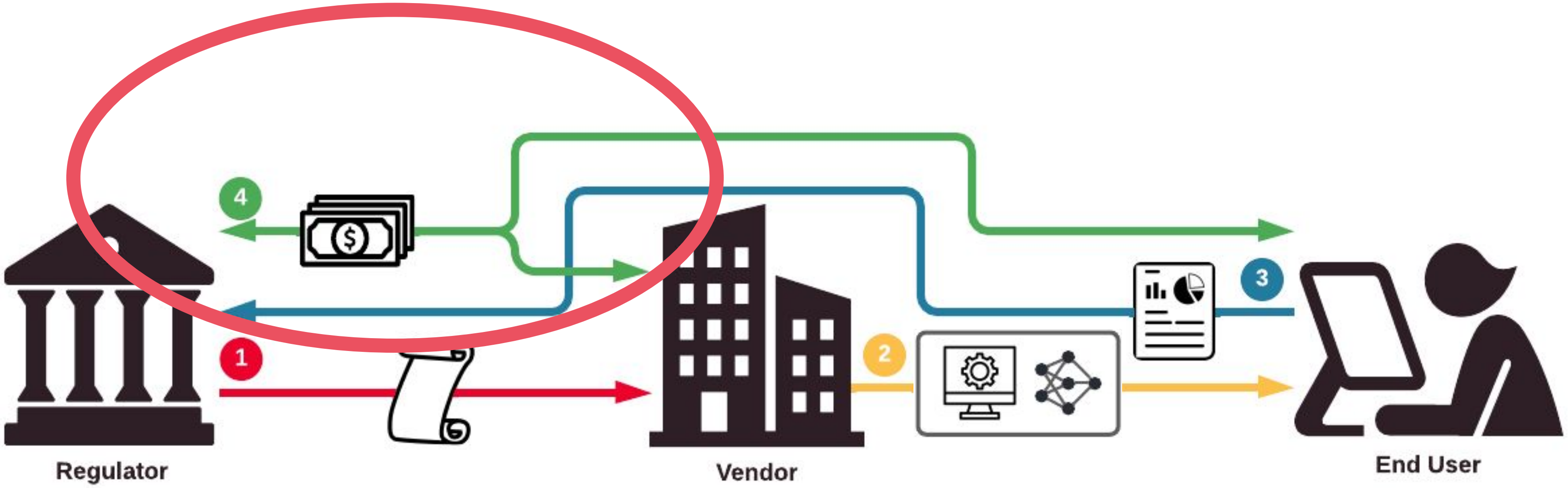
Takeaways

Vendors can compensate users for slowdowns due to hardware patches based on individual use cases

Takeaways

Authorities can incentivize users for running in secure mode!

COMMAND Overview



How much should users receive for running security?

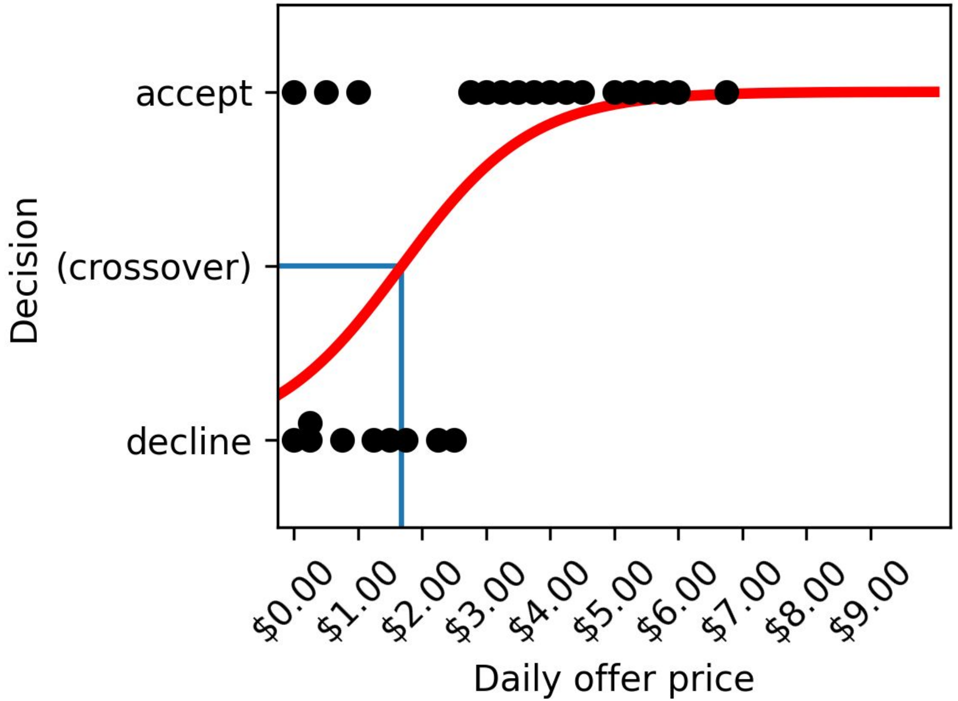
- We conducted an IRB approved user study to obtain answers to this question
  - Our methodology is ‘incentive compatible’ to elicit true responses (as opposed to surveys)

Methodology:

- Participants are offered money to slow down their computer by 10%, 20%, or 30% for 24 hours
- Repeat for 7 days

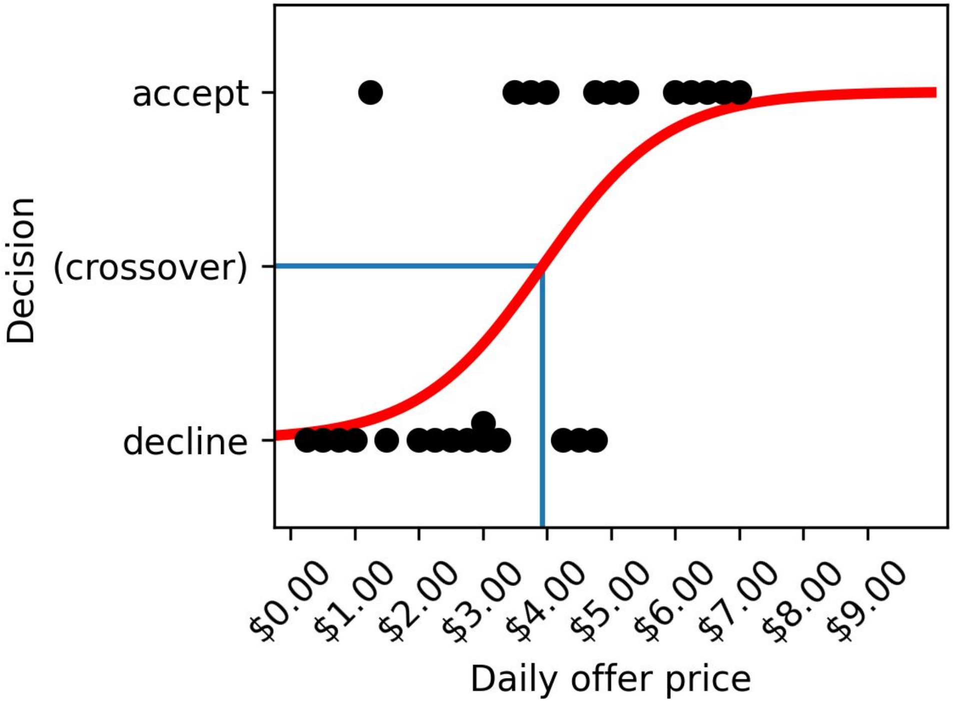
How much do users demand for performance losses?

Willingness to accept a slowdown of **10**% for 24 hours



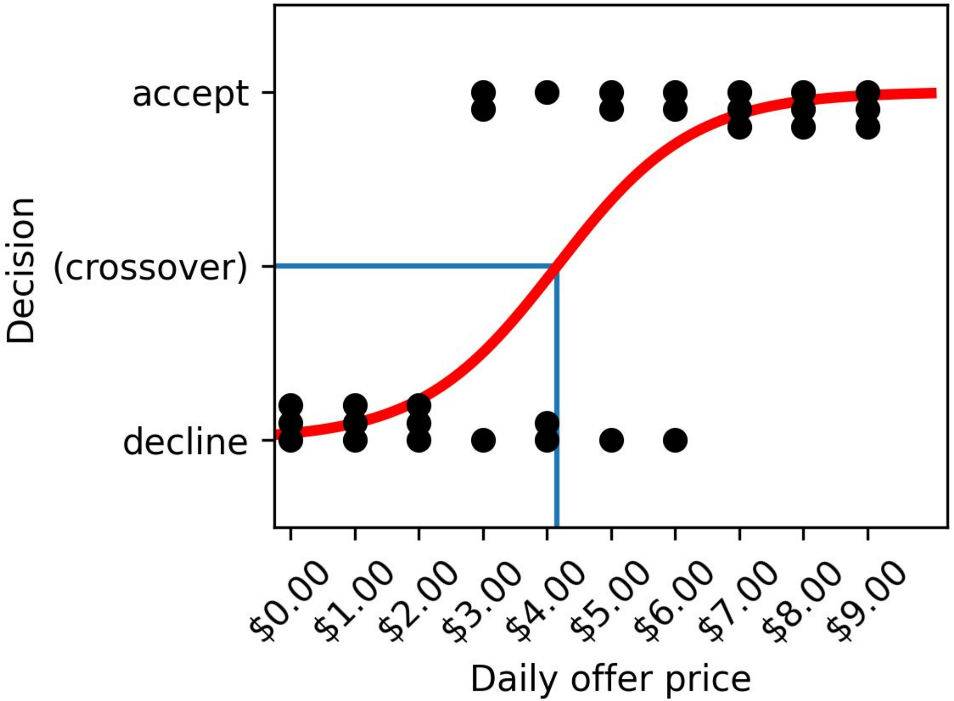
How much do users demand for performance losses?

Willingness to accept a slowdown of **20**% for 24 hours



How much do users demand for performance losses?

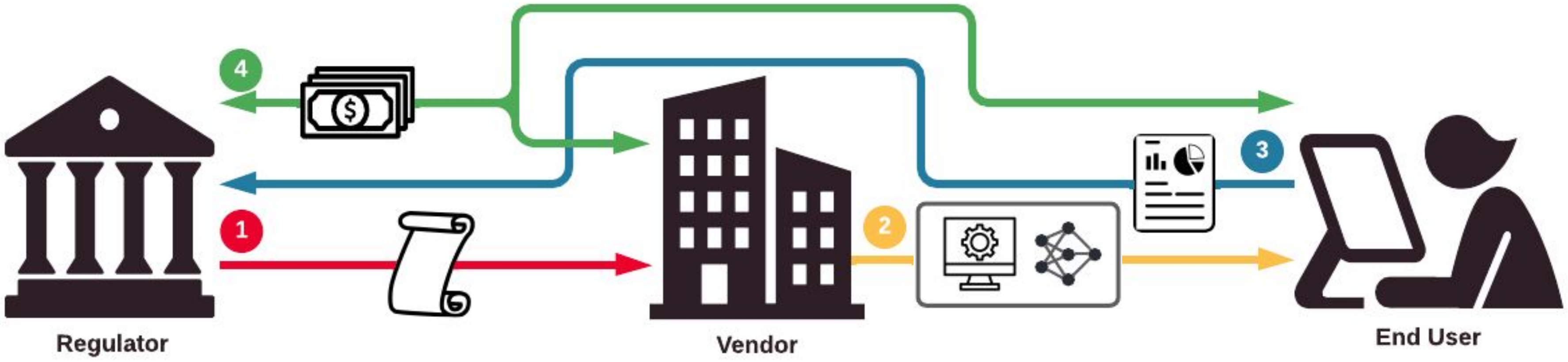
Willingness to accept a slowdown of **30**% for 24 hours



Takeaways

Established a methodology for quantifying $ cost of hardware patches.

COMMAND: A Open Mandates Mechanism



Conclusion

Thank you!

https://arxiv.org/abs/2203.05015
https://arxiv.org/abs/2007.09537

- Problem: Misaligned incentives prevent meaningful security progress
  - Discussed three case studies
  - Motivated the need for equitably sharing burdens
- Solution: a new mechanism design called COMMAND
  - Key idea: *all* vendors set aside certain fraction of costs and resources towards security
  - Described technical mechanisms to enable enforcement and incentivize security adoption
- Looking forward: this is the first step
  - Richer model: How do include insurance and deterrence through punishment in the model?
  - Technical mechanisms for supporting insurance and deterrence?
- Interested in participating?
  - CCC Workshop on Mechanism Design; One page position paper
  - For questions, comments and details please send email to simha@columbia.edu