Cybersecurity: Challenges, Attacks, and Defenses

PROFESSIONAL EDUCATION

CSAIL
MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY

Cybersecurity: Challenges, Attacks, and Defenses

Srini Devadas

Webster Professor of Electrical Engineering and Computer Science

Computer Science and Artificial Intelligence Laboratory (CSAIL)

Massachusetts Institute of Technology

Agenda

Why is security hard?

Threat models

Defensive strategies and examples

- Prevention
- Resilience under attack
- Detection and Recovery

Attacks on Individuals: Ransomware

- Worm enters system through downloaded file.
- Payload encrypts user's hard drive and deletes the original files – user cannot decipher his/her own files
- Pay $500 in Bitcoin to get your files back!

Attacks on Services
Target Store in 2013

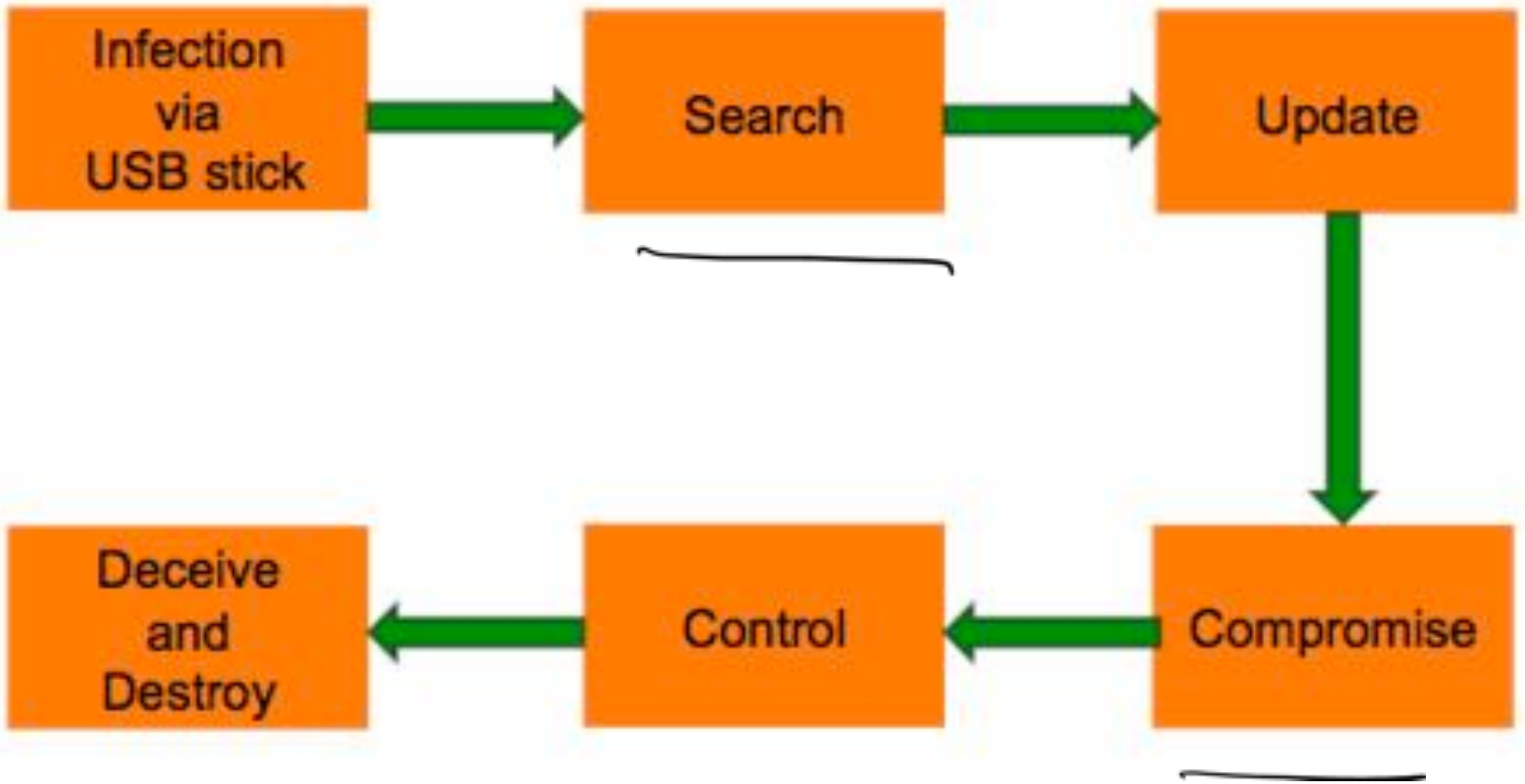
40 million: Number of credit and debit cards thieves stole.
70 million: The number of records stolen that included names, and addresses
46: % drop in profits in the 4th quarter of 2013, compared to 2012.
200 million: Estimated cost for reissuing 21.8 million cards.
53.7 million: The income that hackers likely generated from the sale of 2 million cards

0: Number of customer cards with AVAILABLE hardware security technology that would have stopped the bad guys from stealing

Attacks on Infrastructure
The Stuxnet Cyberphysical Attack

- A 500 Kbyte computer worm that infected the software of at least 14 industrial sites in Iran including a nuclear facility
- Goal was to cause fast-spinning centrifuges to tear themselves apart
- Stuxnet was tracked down by Kaspersky Labs but not before it did some damage

How Stuxnet Worked



Why Do These Attacks Happen? *or* What Makes Security Hard?

Security is a negative goal.

- Want to achieve something despite whatever adversary might do.

Positive goal: "Frans can read grades.txt".

- Ask Frans to check if our system meets this positive goal.

Negative goal: "Nick cannot read grades.txt".

- Ask Nick if he can read grades.txt?
- Must reason about all possible ways in which Nick might get the data.

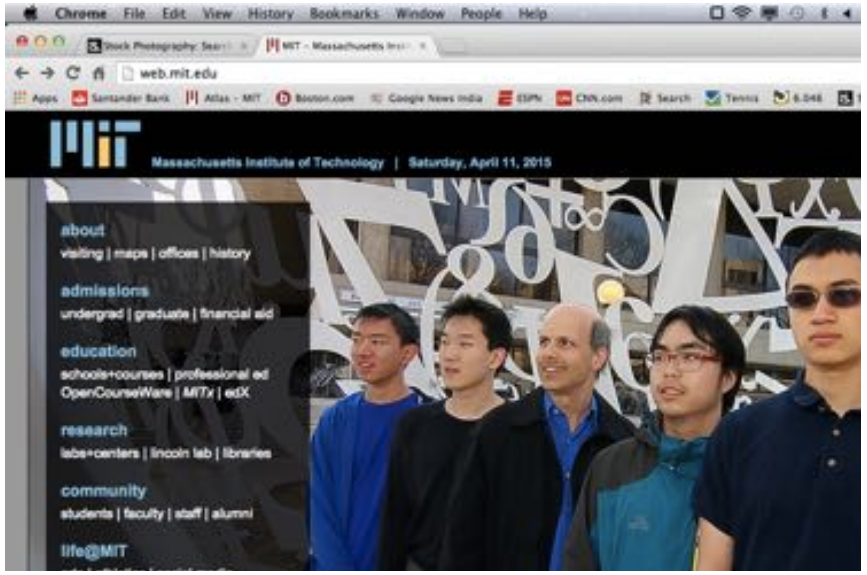
Ways to Access Grades.txt

- Change permissions on grades.txt to get access

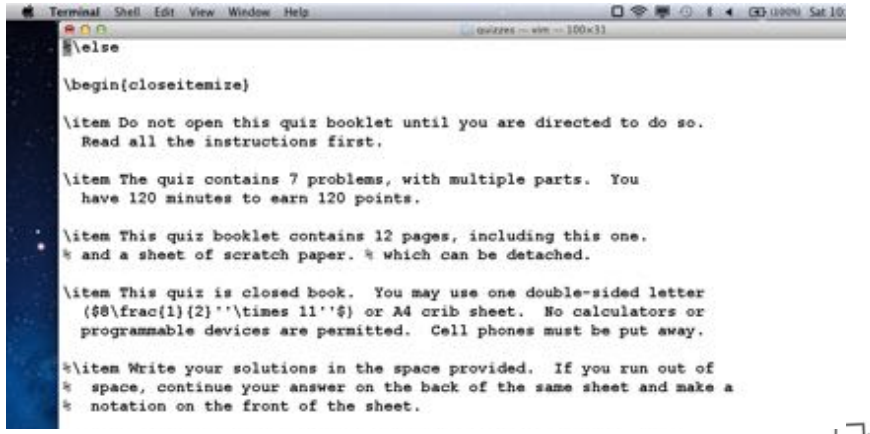
- Access disk blocks directly

Ways to Access Grades.txt

- Access grades.txt via web.mit.edu



- Reuse memory after Frans's text editor exits, read data



Ways to Access Grades.txt

- Read backup copy of grades.txt from Frans's text editor

- Intercept network packets to file server storing grades.txt

Ways to Access Grades.txt

- Send Frans a trojaned text editor that emails out the file

- Steal disk from file server storing grades.txt

Ways to Access Grades.txt

- Get discarded printout of grades.txt from the trash

- Call sysadmin, pretend to be Frans, reset his password

Agenda

Why is security hard?

Threat models

Defensive strategies and examples

- Prevention
- Resilience under attack
- Detection and Recovery

Why Threat Models?

Often don't know in advance who might attack, or what they might do.

- Adversaries may have different goals, techniques, resources, expertise.
- Adversary might be your hardware vendor, software vendor, administrator, ..

Cannot be secure against arbitrary adversaries, as we saw with Nick vs. Frans.

Need to make some plausible assumptions to make progress.

What Does a Threat Model Look Like?

- Adversary controls some computers, networks (but not all).
- Adversary controls some software on computers he doesn't fully control.
- Adversary knows some information, such as passwords or keys (but not all).
- Adversary knows about bugs in your software

What Does a Threat Model Look Like? – 2

Physical attacks?

Social engineering attacks?

Many systems compromised due to unrealistic / incomplete threat models.

- Adversary is outside of the company firewall
- Adversary doesn't know legitimate users' passwords.
- Adversary won't figure out how the system works.

Cybersecurity and Threats

Cybersecurity is a property of computer systems similar to performance and energy

Attackers take a holistic view by attacking any component or interface of system

Diverse threat models dictate different desirable security properties

One Philosophy

- Computer systems are so complex that it is impossible to design them without vulnerabilities.
- Best strategy is therefore to:
  - Focus on existing computing systems and their attacks to discover flaws
  - Design mechanisms into systems to protect against these attacks
  - Manage risk and administer systems well

One Philosophy

Unfortunately, new flaws are always discovered…

Can we do better than a “Patch & Pray, Perimeter Protection” mindset?

A Holistic Philosophy

Security property cannot be articulated well when *isolated* to a component or layer

→ need a systems-wide, architectural viewpoint

New theoretical and practical foundations of secure computing that integrate security in the design process

→ security "by default"

→Remove program error as a source of vulnerability

Three Defenses

Prevention: Increasing the difficulty of attacks

Resilience: Allowing a system to remain functional despite attacks

Detection and Recovery: Allowing systems to more quickly detect and recover from attacks to fully functional state.

Agenda

Why is security hard?

Threat models

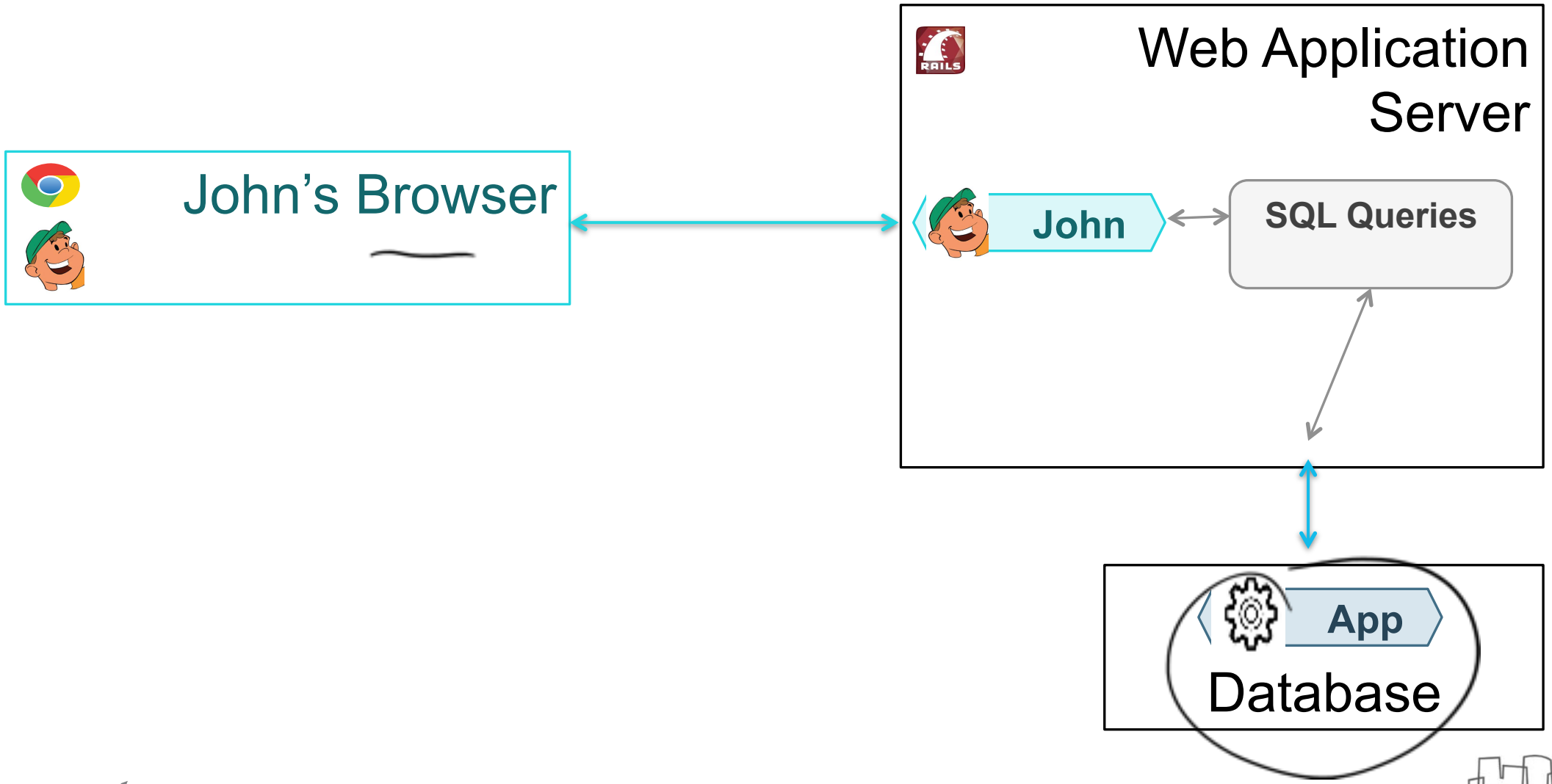
Defensive strategies and examples

- Prevention
- Resilience under attack
- Detection and Recovery

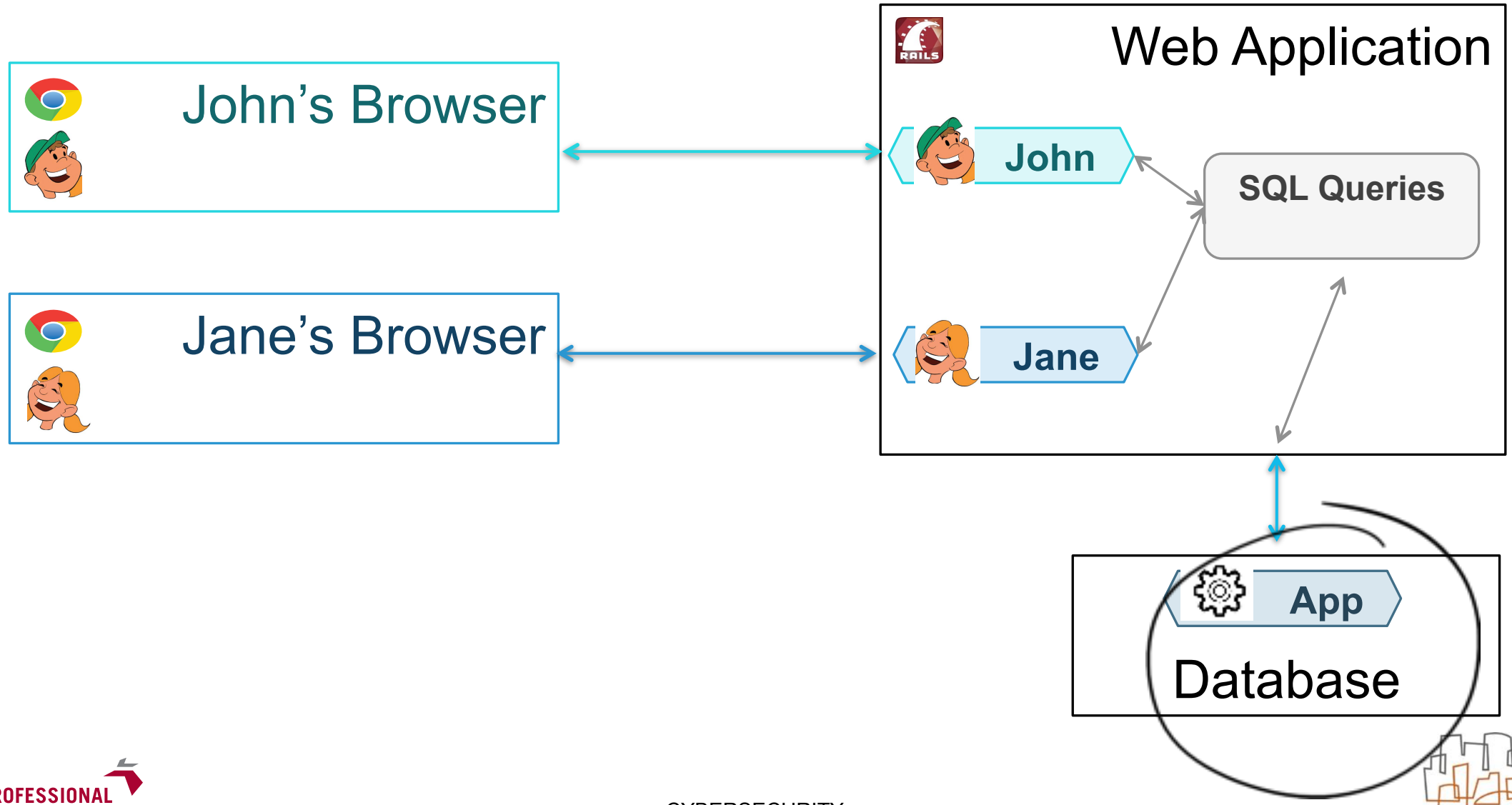
Prevention

Information Flow Tracking for Web application security

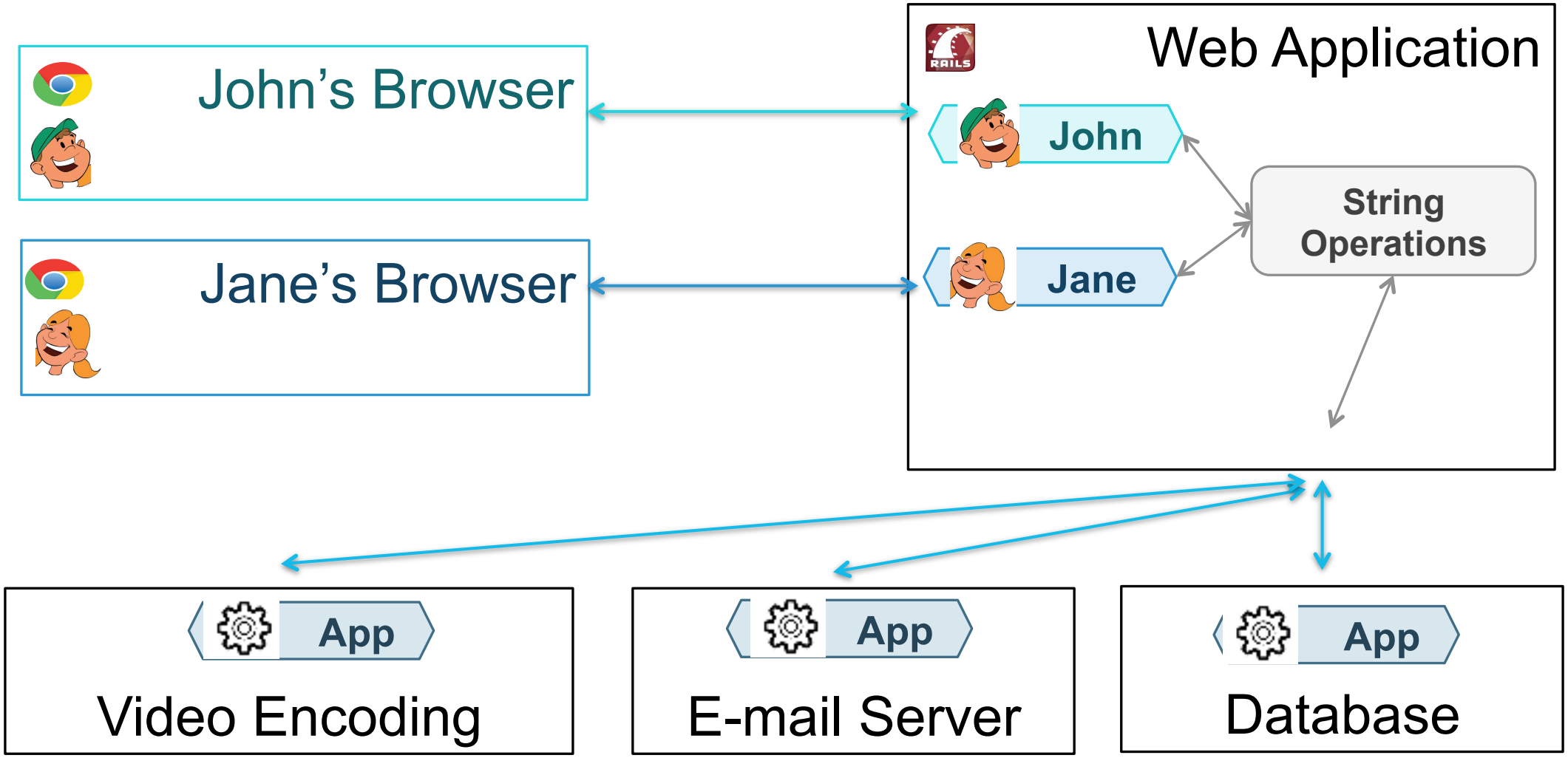
Deputy Confusion in Web Applications



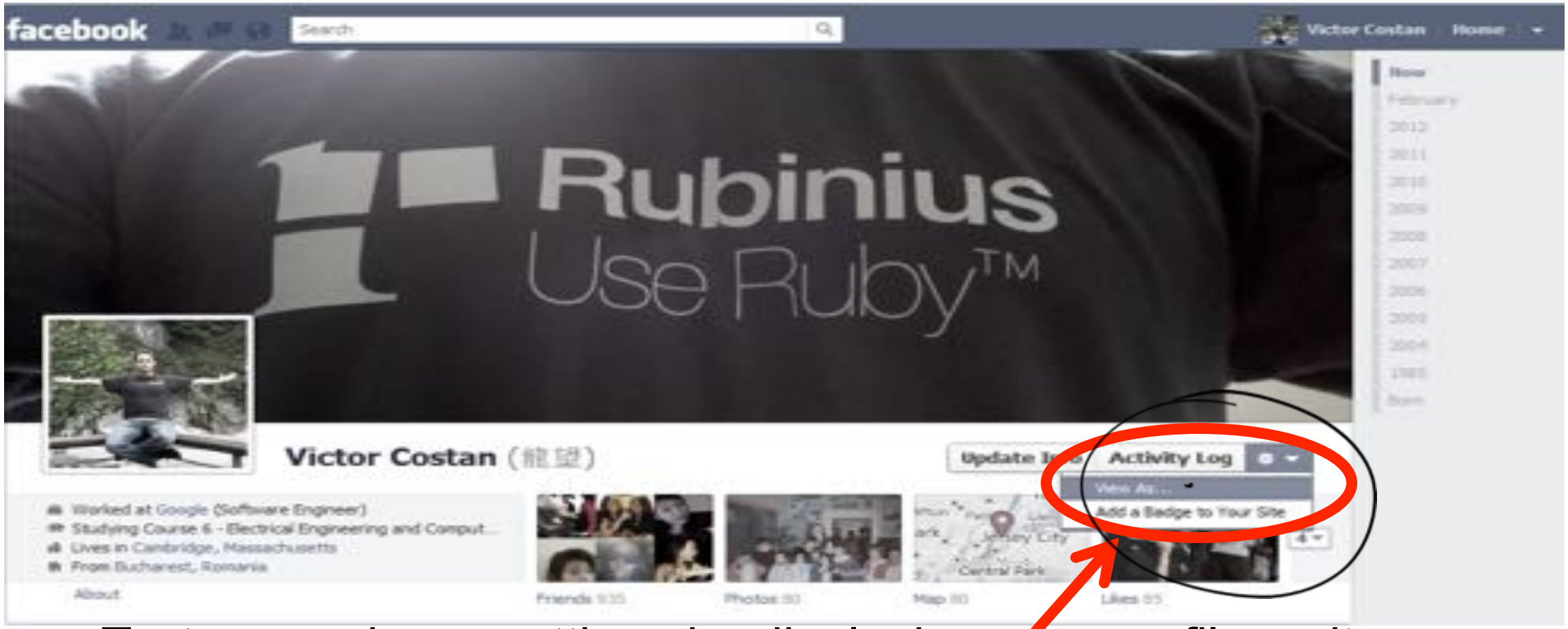
Deputy Confusion in Web Applications



Deputy Confusion in Web Applications



Deputy Confusion at Facebook



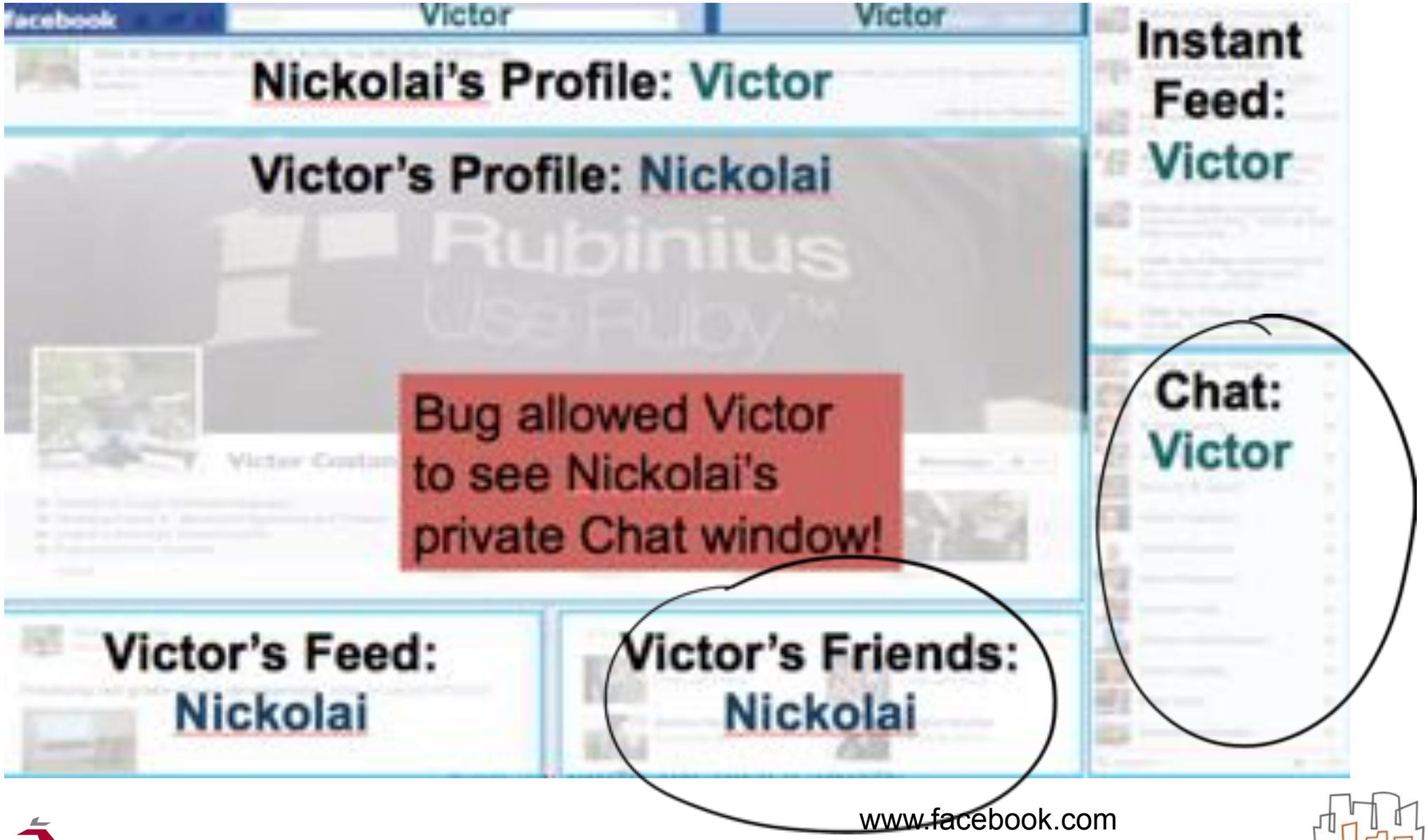
Test your privacy settings by displaying your profile as it is shown to your friends

www.facebook.com

Deputy Confusion at Facebook

www.facebook.com

Deputy Confusion at Facebook



www.facebook.com

Deputy Confusion at Facebook

www.facebook.com

Deputy Confusion at Facebook

Facebook Chat Down for Maintenance Following Privacy Lapse

May 05, 2010 by Jennifer Van Grove

"Facebook Chat is now down for maintenance. The feature was presumably disabled following a report that exposed a Facebook security bug that allowed users to access and view friends' live chats, friend requests and friends in common.

The report indicates that access to this personal information was accessible via Facebook's privacy settings, with the Preview My Profile feature creating the loophole to access the private live chats of friends.

With Preview My Profile, users can view how their profile appears to any given Facebook friend. The bug apparently let those users see the live chats and friend requests of the friend in question.

Unfortunately for the company, this is not the first time users' personal information has been exposed without consent. Earlier this year, user e-mail addresses were exposed in a hiccup following a site update."

Eliminate the Confusion!

Add Labels to Data

Labels address deputy confusion


This text was typed by Victor

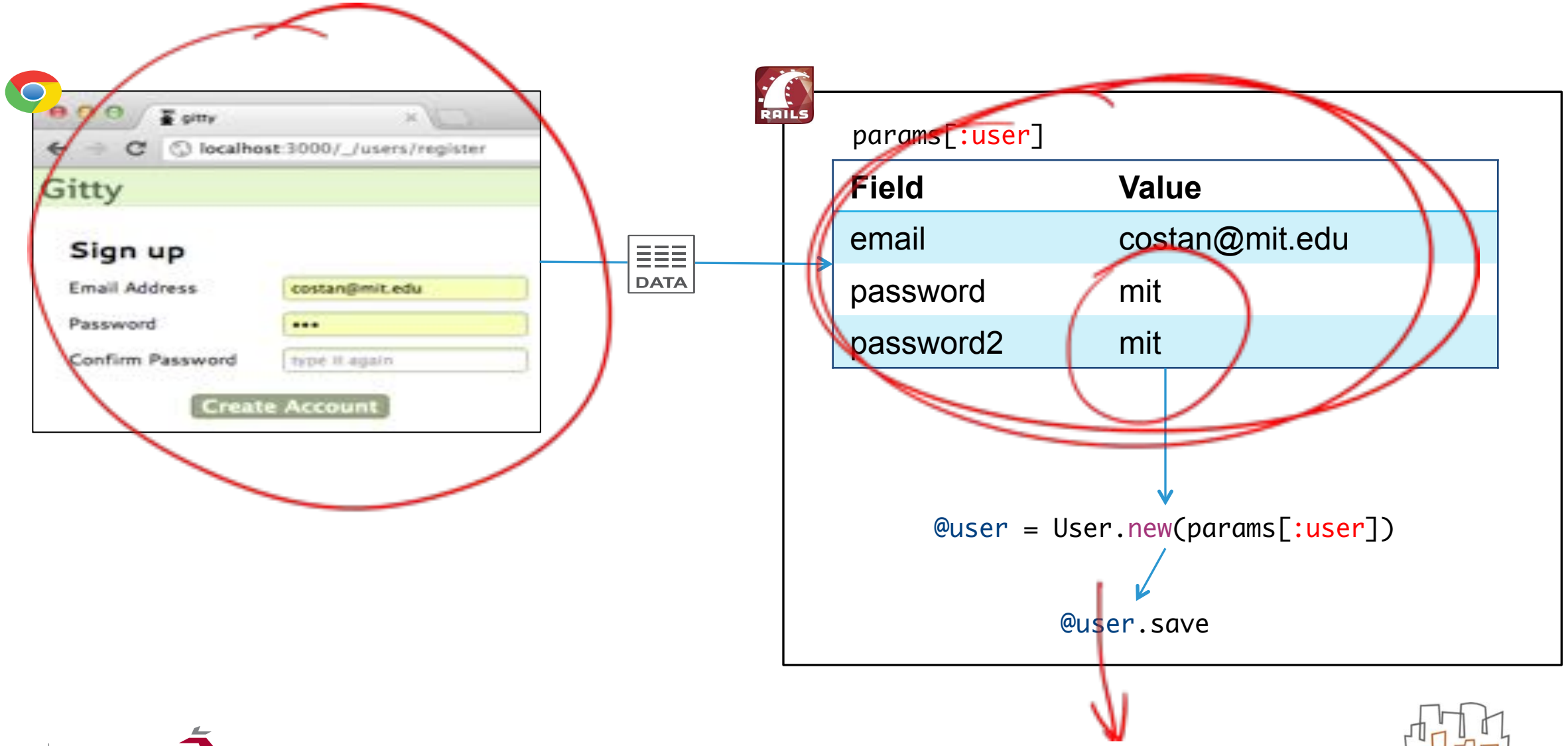

Only show this to Victor's friends

Filter Output Data

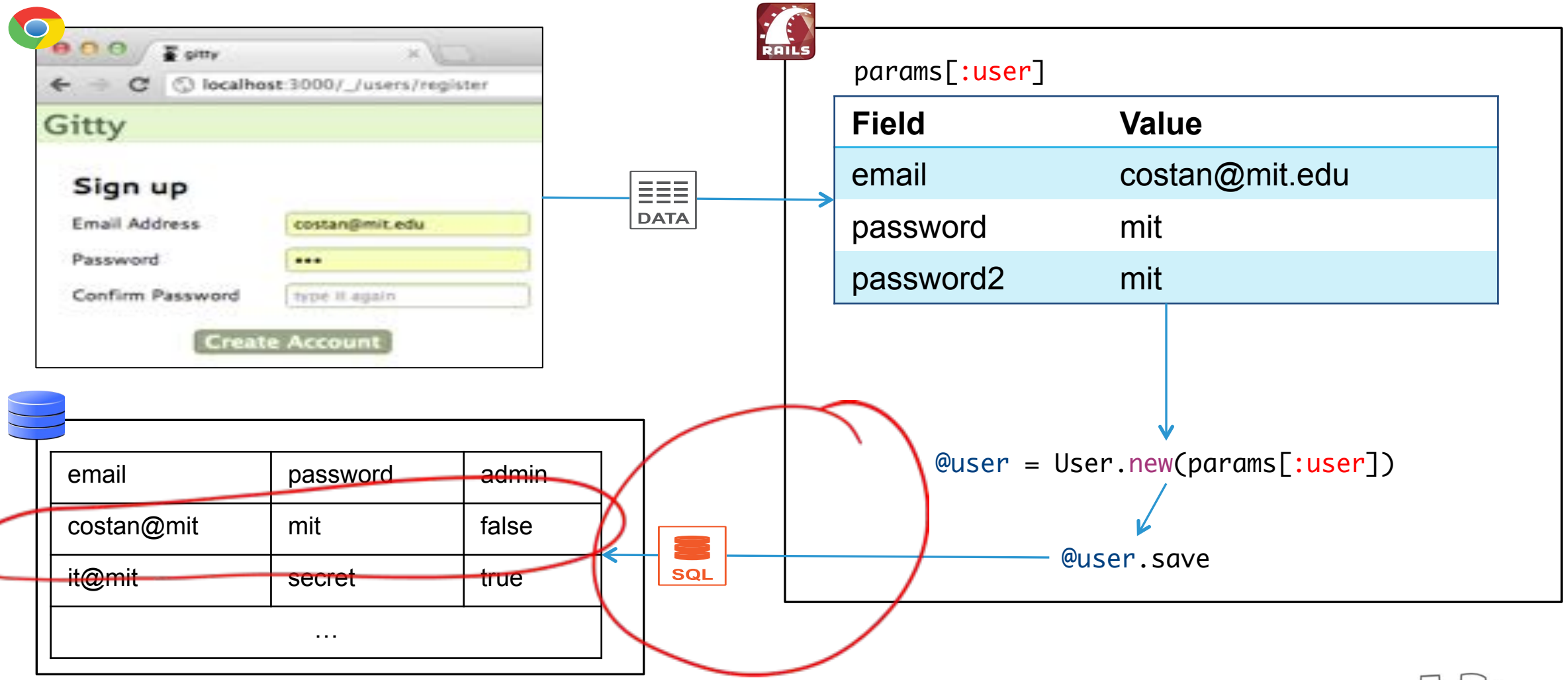
Prevent deputy confusion

- Check security policies before making database changes
- Check privacy policies before outputting data to the user

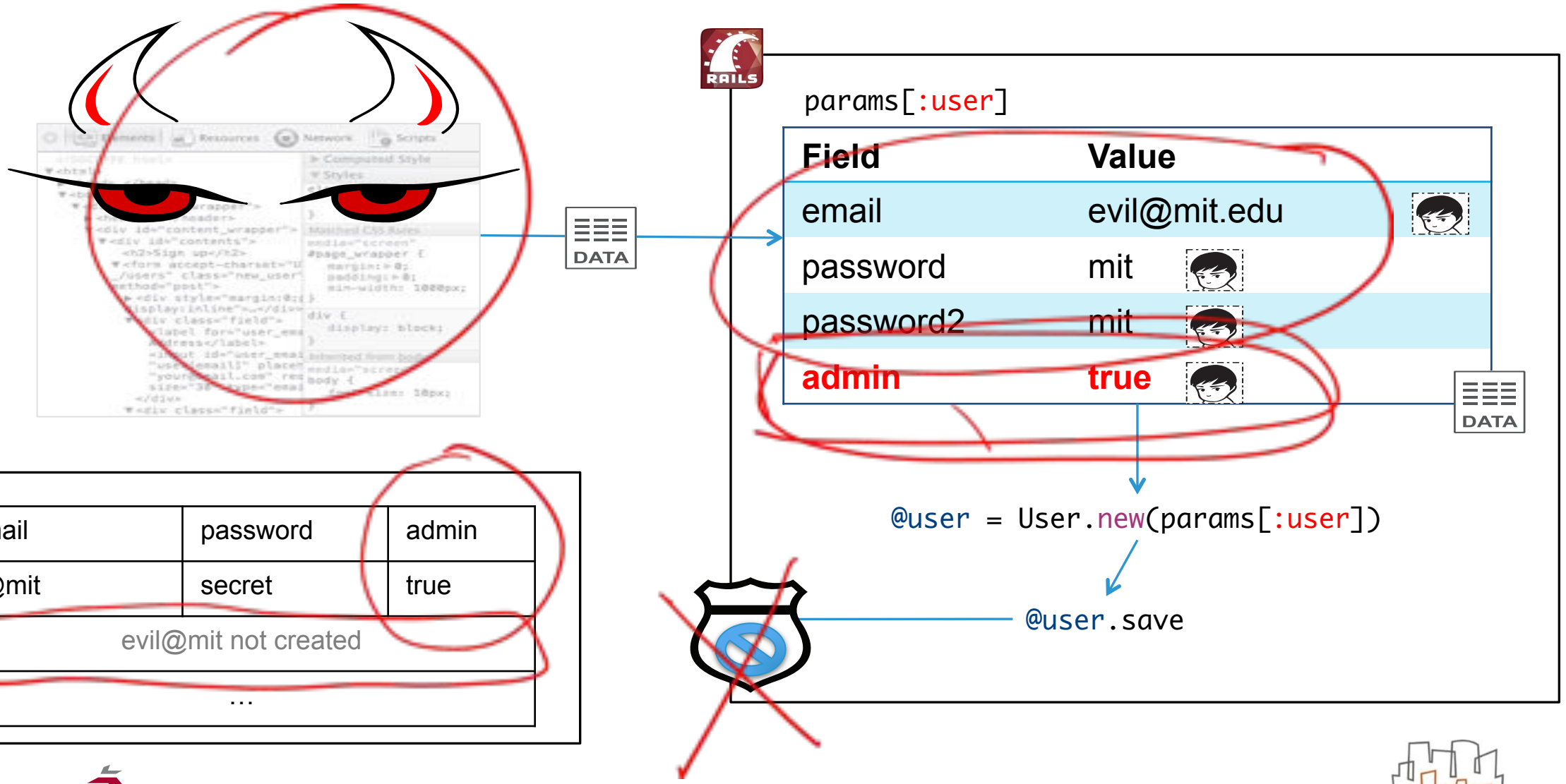
Web Application without Labels



Web Application without Labels



Eliminating Deputy Confusion



Eliminating Deputy Confusion

Field	Policy
email	Users can edit their own
password	Users can edit their own
admin	Admins can edit any

email	password	admin
costan@mit	mit	false
it@mit	secret	true
…		

RAILS

params[:user]

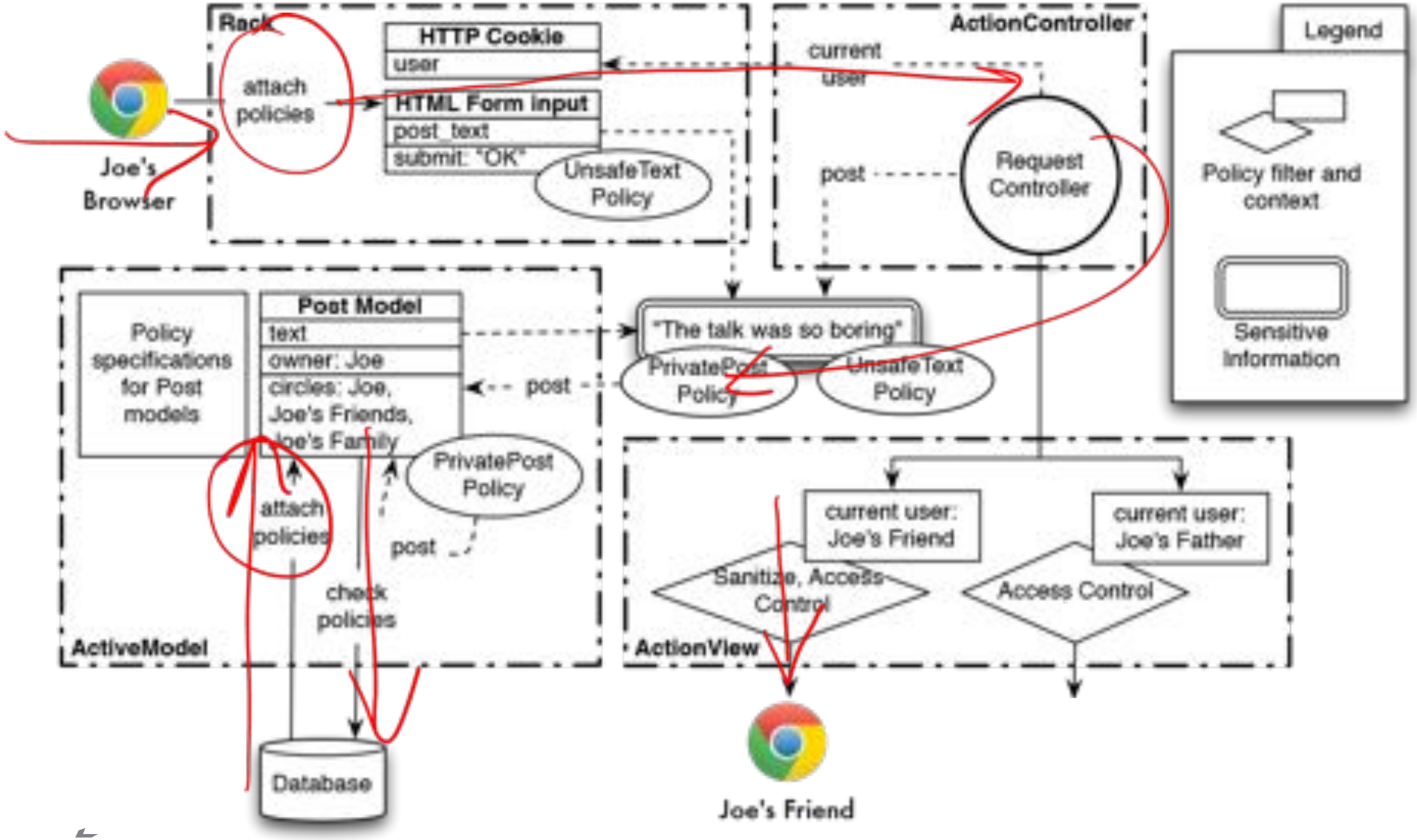
Field	Value
email	costan@mit.edu
password	mit
password2	mit
admin	true

DATA

```
@user = User.new(params[:user])
```

```
@user.save
```

Resin on Rails



Resilience Under Attack

Encrypted Computation

Trusted Computing Base

The trusted computing base (TCB) is the set of software and hardware components that need to be trusted by a user

In TPM-based systems, the TPM attests the veracity of several millions of lines of (buggy) OS code

- The TPM does not provide real security

In the cloud, the TCB is > 20M lines of code from tens of software vendors

- No wonder we have so many security breaks!

Trusted Computing Base

The trusted computing base (TCB) i… f software and hardware components that … ed by a user

In TPM-based system… he veracity of several millions of … code

– The TPM … ecurity

In the clou… OM lines of code from tens of software ve…

– No wond… have so many security breaks!

CAN WE SHRINK THE TCB?

Application: Secure Cloud Computing

I want to delegate processing of my data, without giving away access to it.

Separating processing from access via encryption:

- I will encrypt my data before sending it to the cloud
- They will apply their processing on the encrypted data, send me processed (still encrypted) result
- I will decrypt the result and get my answer

Computation Under Encryption

An Analogy: Alice's Jewelry Store

- Alice's workers need to assemble raw materials into jewelry

An Analogy: Alice's Jewelry Store

- Alice's workers need to assemble raw materials into jewelry

An Analogy: Alice's Jewelry Store

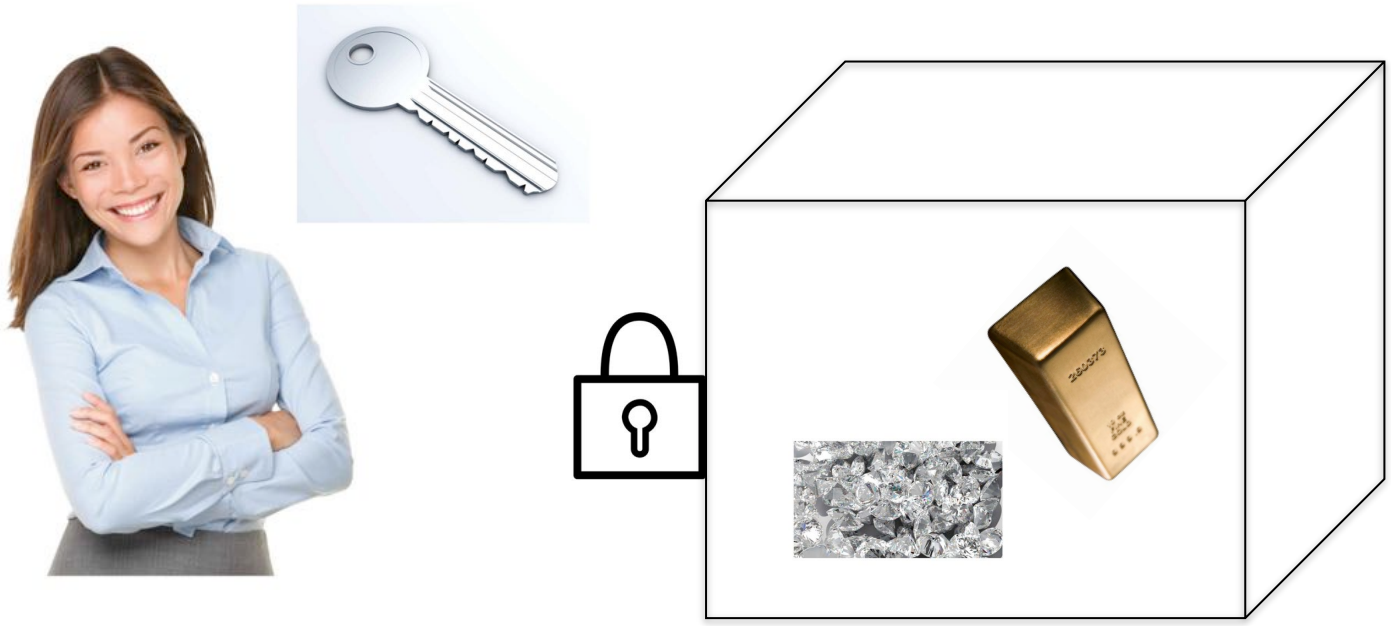
- Alice's workers need to assemble raw materials into jewelry
- But Alice is worried about theft

An Analogy: Alice's Jewelry Store

- Alice's workers need to assemble raw materials into jewelry
- But Alice is worried about theft
  - How can the workers process the raw materials without having access to them?

An Analogy: Alice's Jewelry Store

- Alice puts materials in locked glove box
  - For which only she has the key

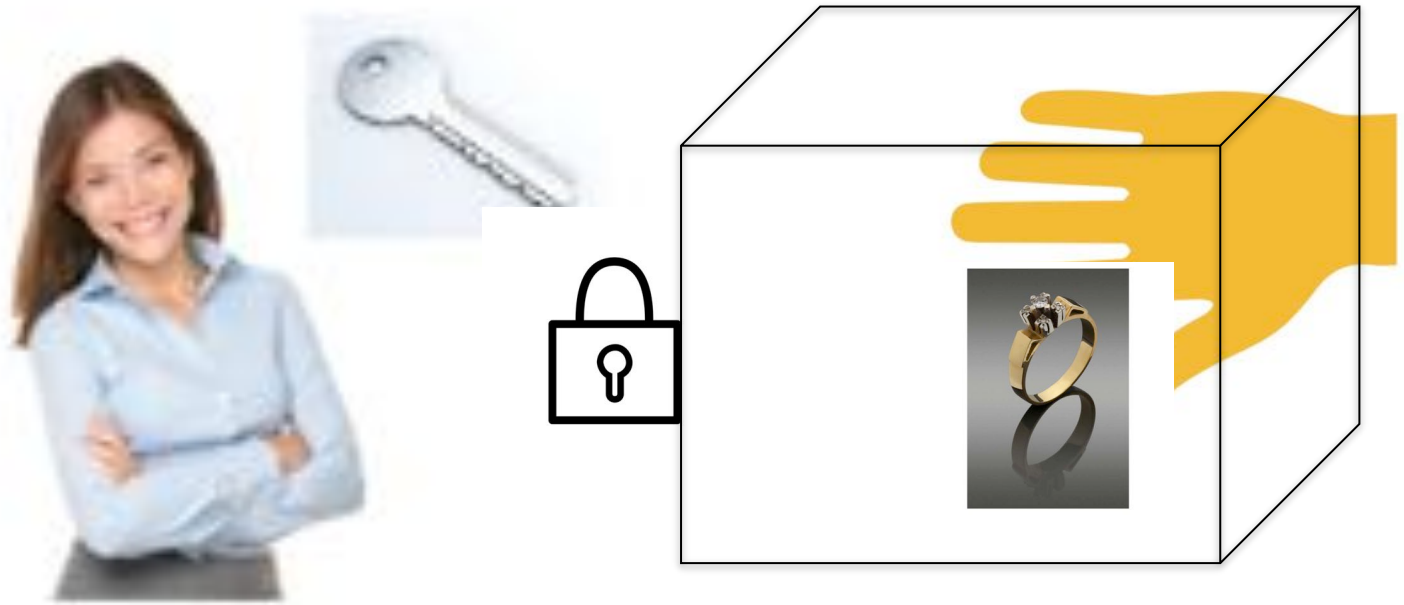


An Analogy: Alice's Jewelry Store

- Alice puts materials in locked glove box
  - For which only she has the key
- Workers assemble jewelry in the box

An Analogy: Alice's Jewelry Store

- Alice puts materials in locked glove box
  - For which only she has the key
- Workers assemble jewelry in the box



An Analogy: Alice's Jewelry Store

- Alice puts materials in locked glove box
  - For which only she has the key
- Workers assemble jewelry in the box
- Alice unlocks box to get "results"

The Analogy

Encrypt: putting things inside the box

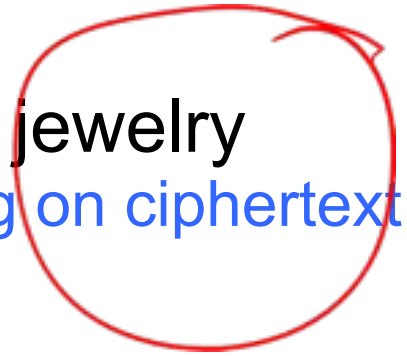
- Alice does this using her key
- $c_i \leftarrow Enc(m_i)$

Decrypt: Taking things out of the box

- Only Alice can do it, requires the key
- $m^* \leftarrow Dec(c^*)$

Process: Assembling the jewelry

- Anyone can do it, computing on ciphertext
- $c^* \leftarrow Process(c_1, \ldots, c_n)$



$m^* = Dec(c^*)$ "the ring", made from "raw material" m_i

Encrypted Computation

Encrypted computation can thus be achieved using Fully Homomorphic Encryption (FHE) **without trusting anything on the server side**

- Server does not need to store a secret key

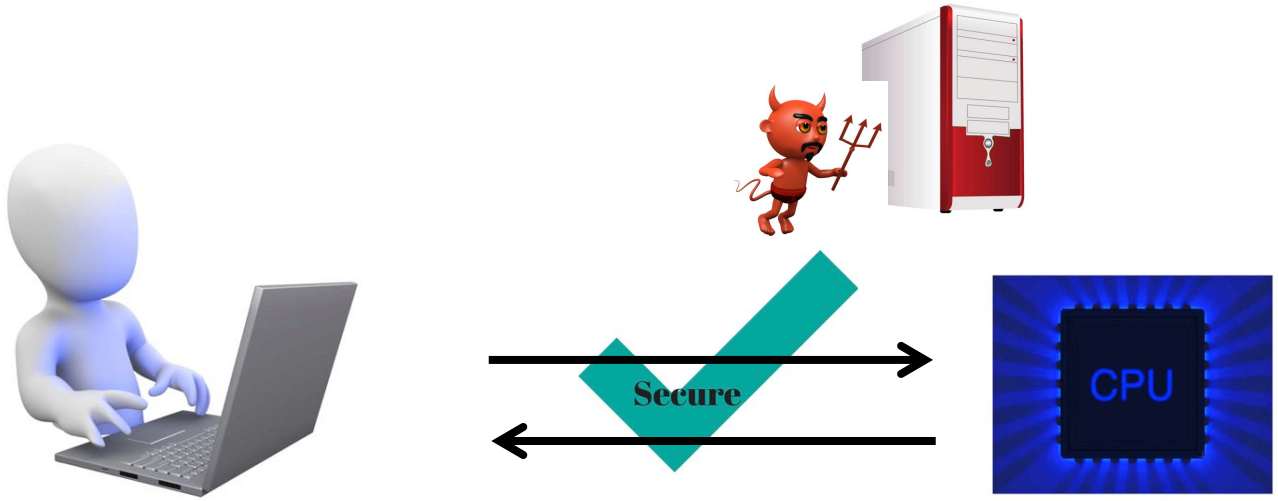
Unfortunately, FHE overheads are about 10^8 to 10^9 for straight-line code and overheads grow if there is complex control in the program

- Only usable for simple computations

What About Hardware Approaches?

Tamper Resistant Hardware

- The secure processor is trusted, shares secret key with client.
- Private information stored in the hardware is not accessible through external means.
- examples: XOM, Aegis, TPM, TPM + TXT, etc.



A Typical Computer TCB

Software…

… Running on hardware

(Ring 3)

App | App | Secure App

Privelege

TRUSTED

OS Kernel (Ring 0)

Hypervisor (Ring 0, VMX root)

BIOS (SMM)

TRUSTED

DRAM

DRAM Ctrl.

CPU

Thread

L1 $

L2 $

L3 $

Main board

Chipset

Disk

Network

Intel SGX to reduce TCB

-SGX protects a small codebase

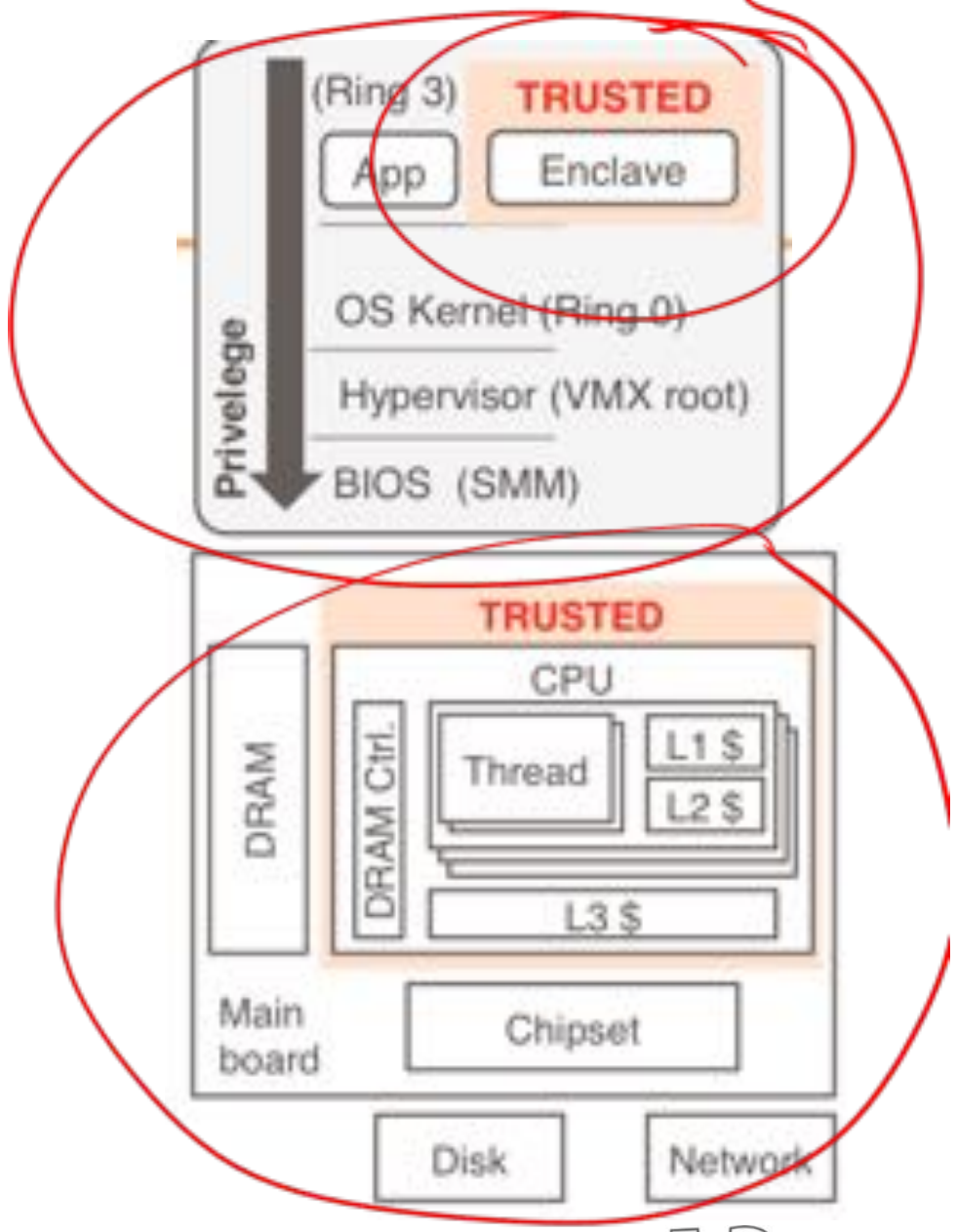
- Doesn't trust OS

- Protected app = *"Enclave"*

-Provides a trusted environment:

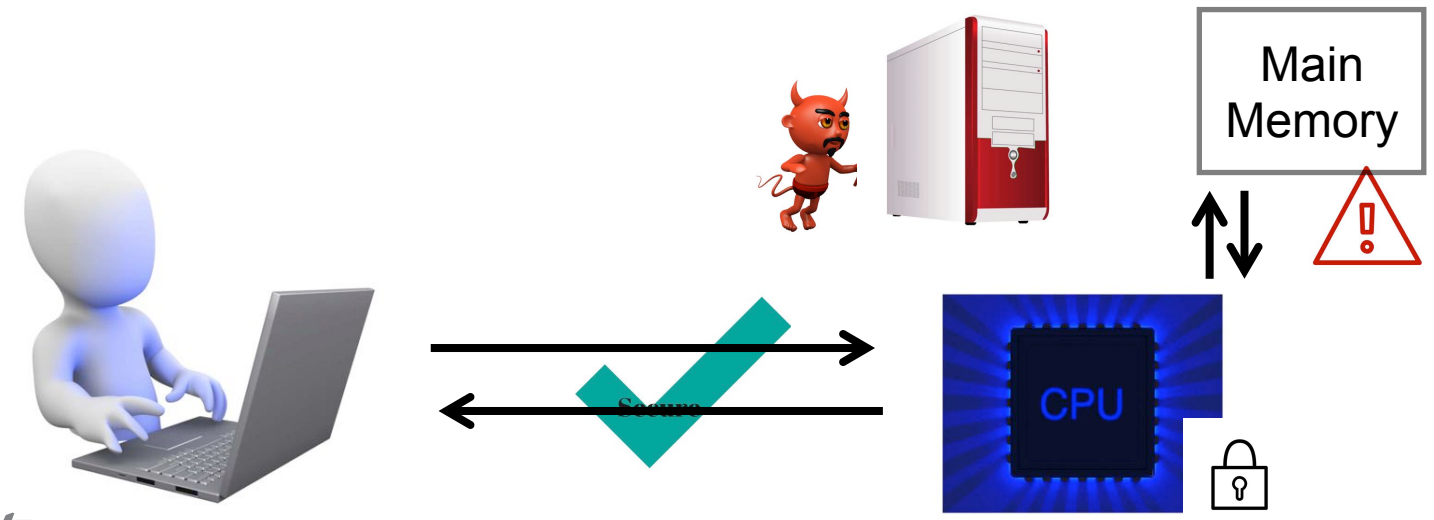
- App integrity
- Protects data

- TCB is the Intel CPU- no off-chip interfaces to secure

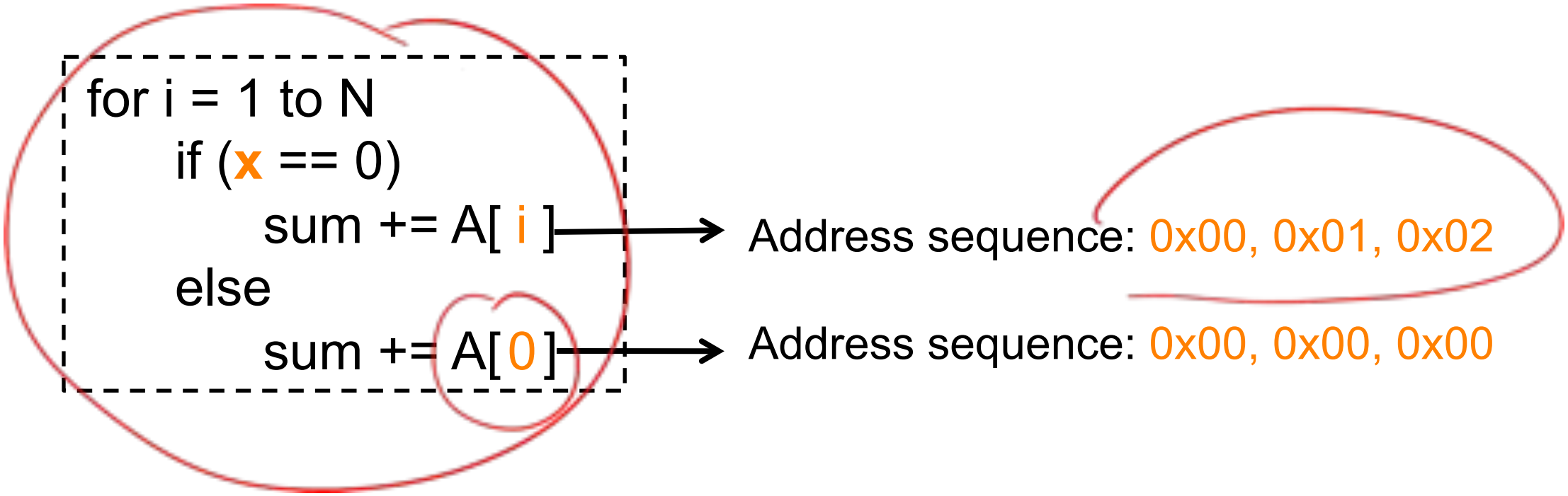


Tamper Resistant Hardware Limitations

- Just trusting the tamper resistance of the chip not enough!
- I/O channels of the secure processor can be monitored by software and leak information
- Examples: address channel, I/O timing channel
- Malicious OS software can monitor these channels



Leakage through Address Channel

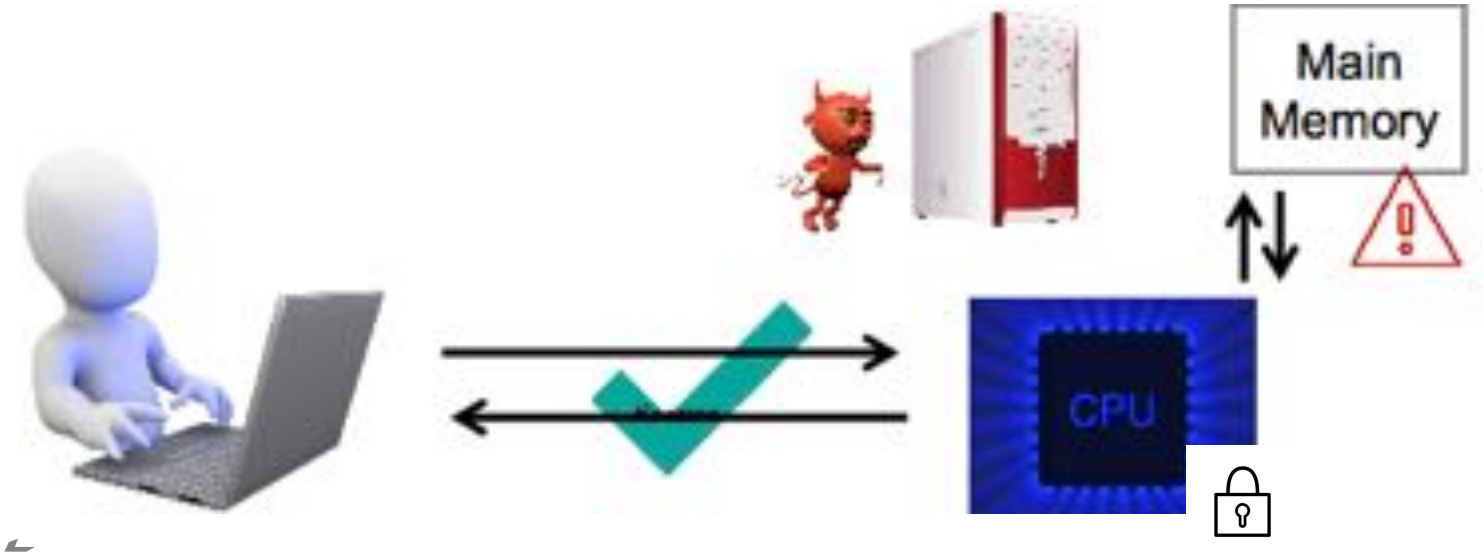


- The value of x is leaked through the access pattern
- Sensitive data exposed by observing the addresses!

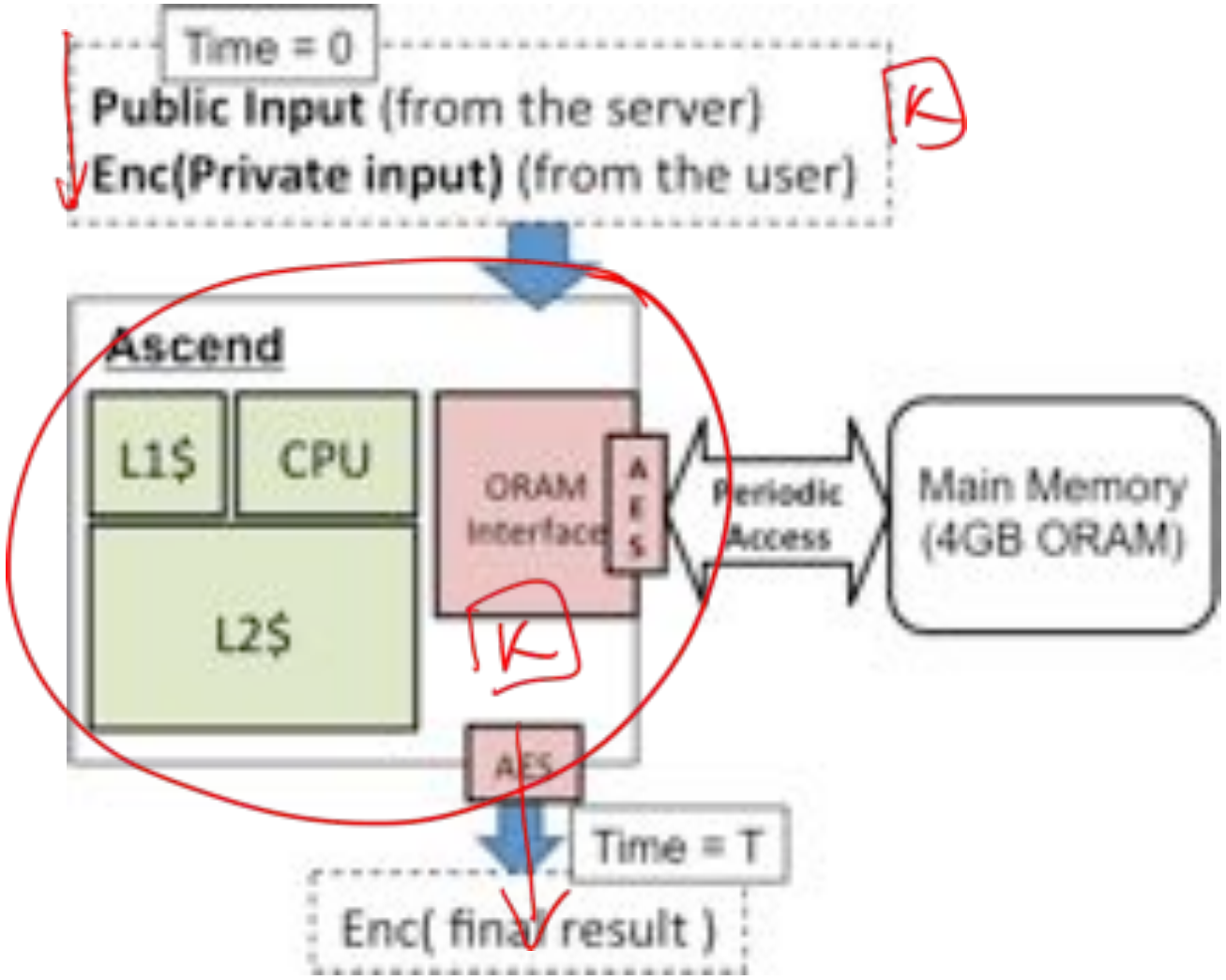
Ascend Processor Security Goal

Protect against all software-based and some hardware-based attacks when running untrusted software

An adversary cannot learn a user's private information by observing the pin traffic of Ascend.



Ascend Processor:
Eliminate leakage over chip pins



Detection and Recovery

Selective Re-Execution

Attackers routinely compromise system integrity

www.wired.com
www.thecrimson.com

Compromises inevitable

- Difficult to write bug-free software
- Administrators mis-configure policies
- Users choose weak, guessable passwords

- Need both “proactive” security, *and* “reactive” recovery mechanisms

Limited existing recovery tools

- Anti-virus tools
  - Only repair for predictable attacks
- Backup tools
  - Restoring from backup discards *all* changes
- Administrators spend days or weeks manually tracking down all effects of the attack
  - No guarantee if they found everything

Challenge: disentangle changes by *attacker* and *legitimate* user

- Adversary could have modified many files directly
- Legitimate processes may have been affected
  - Users ran trojaned pdflatex or ls (listing files)
  - SSH server read a modified /etc/passwd
- Those processes are now suspect as well

Retro: help users disentangle on one machine

- Record history of all computations on machine
- After intrusion found, roll back affected objects
- Re-execute actions that were indirectly affected
- *Minimize* user input required to disentangle
  - User edited attacker's file with emacs (can't replay everything)
  - External effects outside of our control (can't undo spam)

Example attack scenario

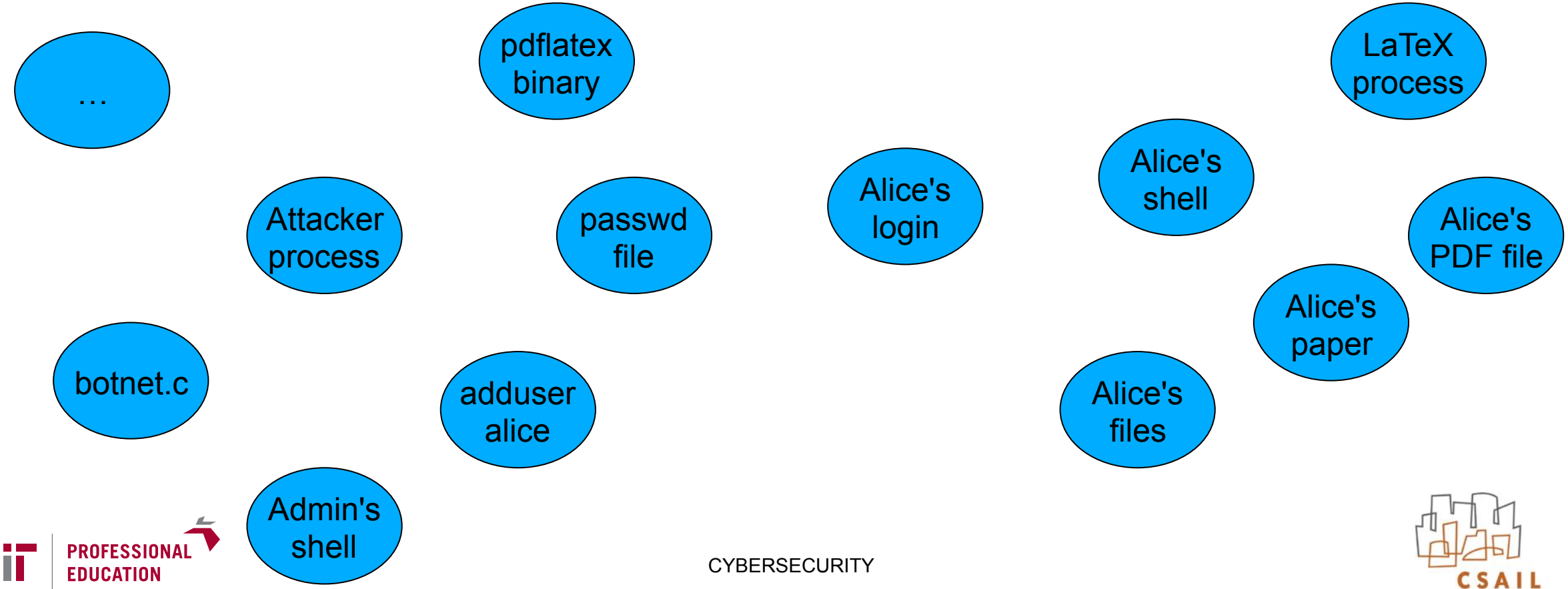
- Attacker modifies /etc/passwd to add new account
- Installs trojan pdflatex, ls to restart, hide botnet

- Admin modifies /etc/passwd to add account for Alice

- Alice logs in via SSH
- SSH server reads /etc/passwd
- Alice runs trojaned pdflatex, ls

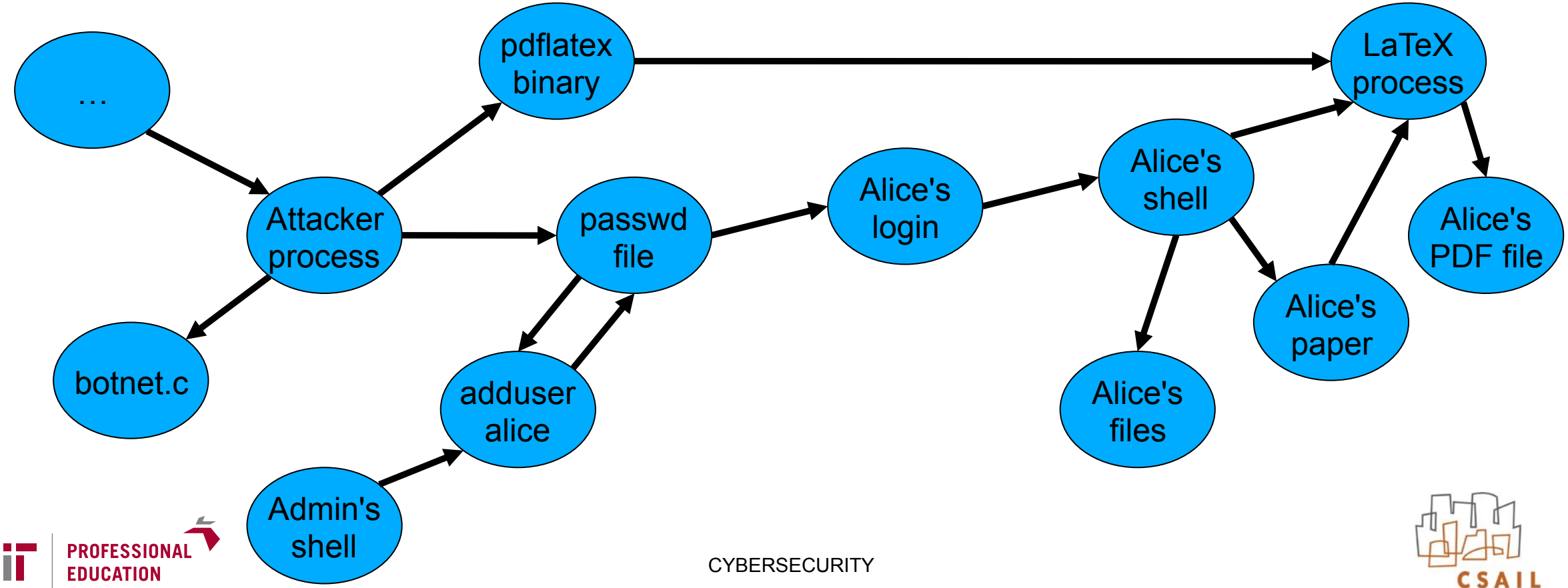
Attacker is not targeting Alice, wants to run botnet

Strawman 1: Taint tracking



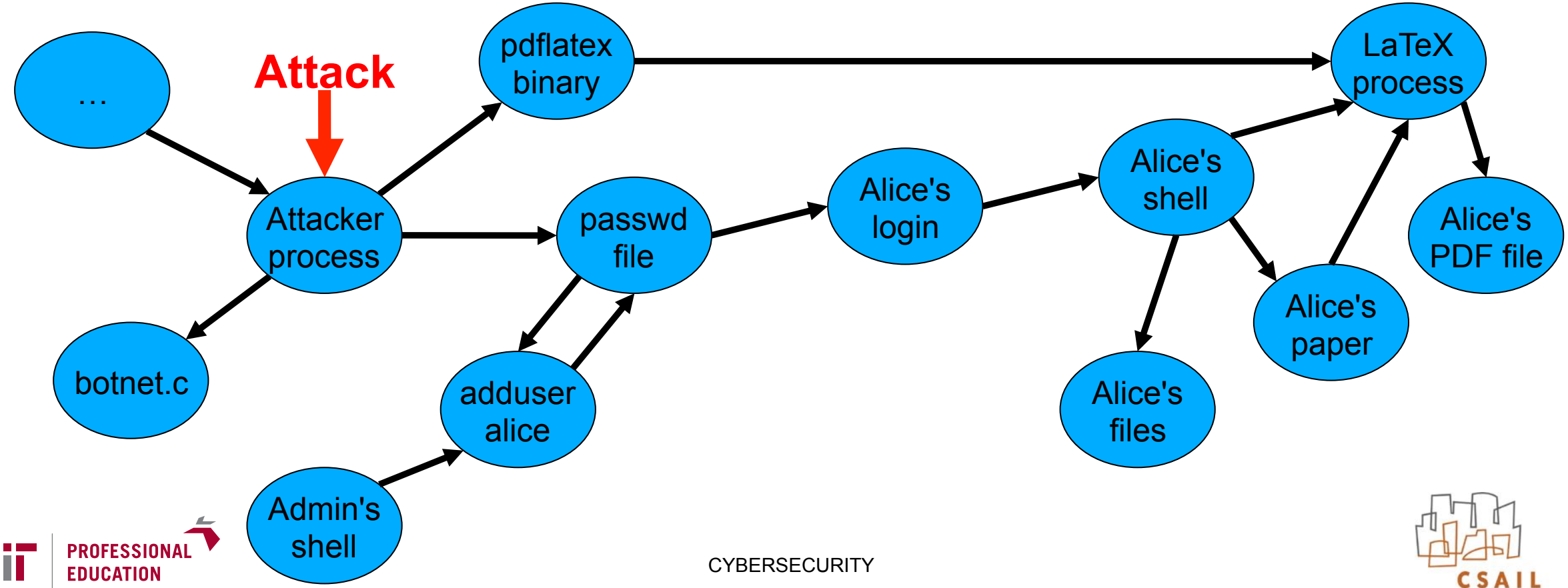
Strawman 1: Taint tracking

- Log all OS-level dependencies in system



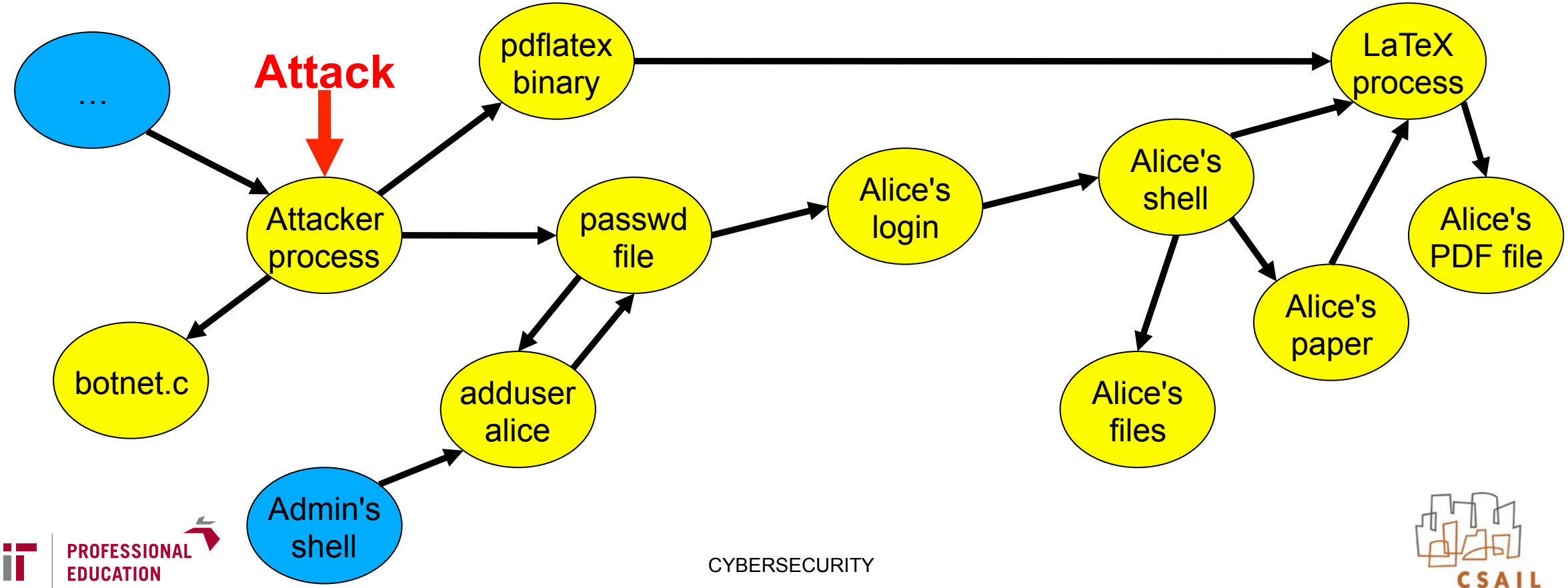
Strawman 1: Taint tracking

- Given attack, track down all affected files, and restore just those files from backup



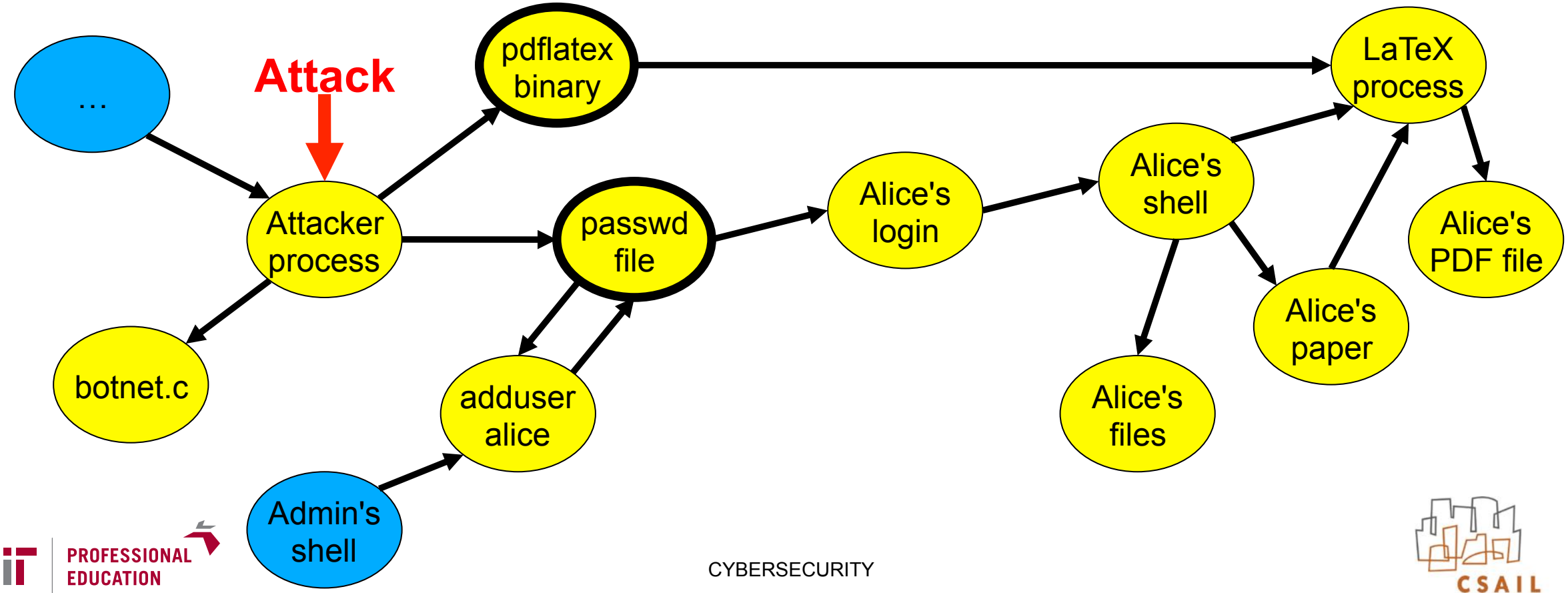
Strawman 1: Taint tracking

- Given attack, track down all affected files, and restore just those files from backup



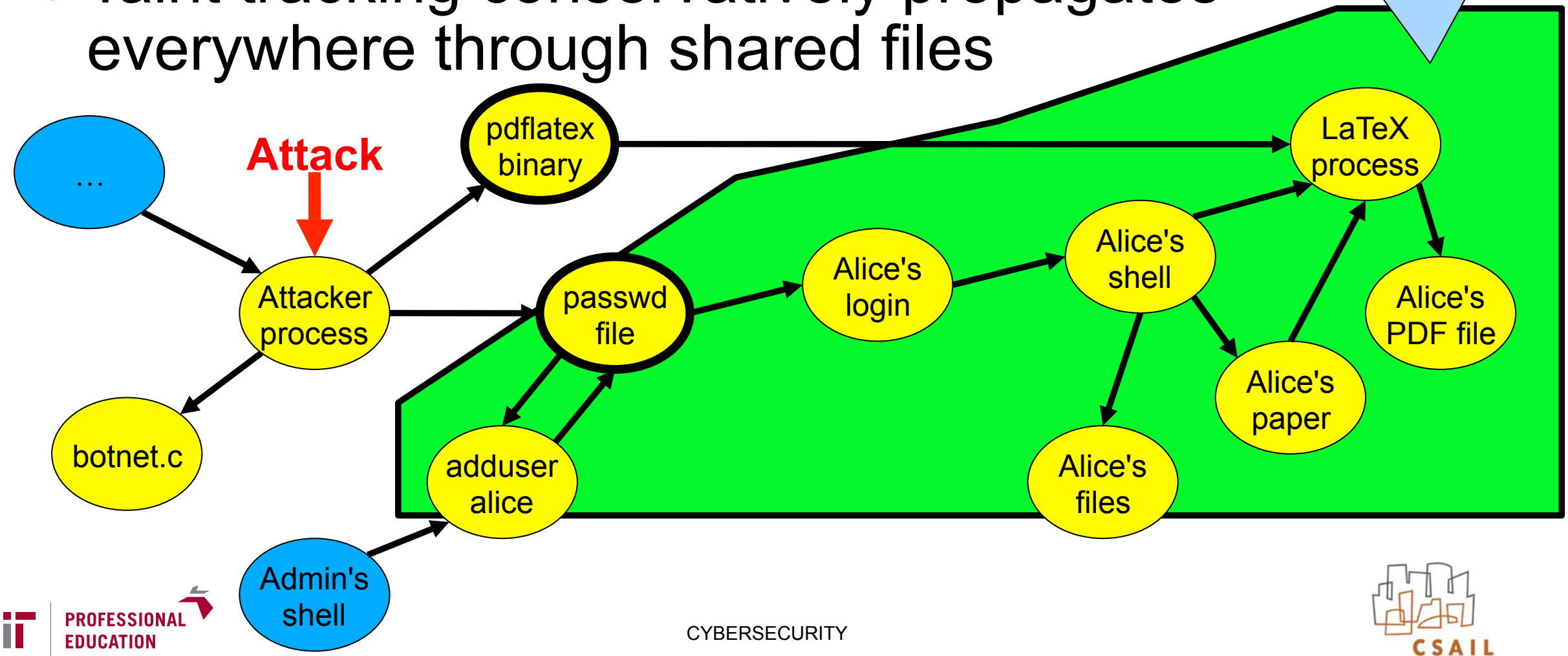
Problem with taint tracking: false positives

- Taint tracking conservatively propagates everywhere through shared files



Problem with taint tracking: fals

- Taint tracking conservatively propagates everywhere through shared files

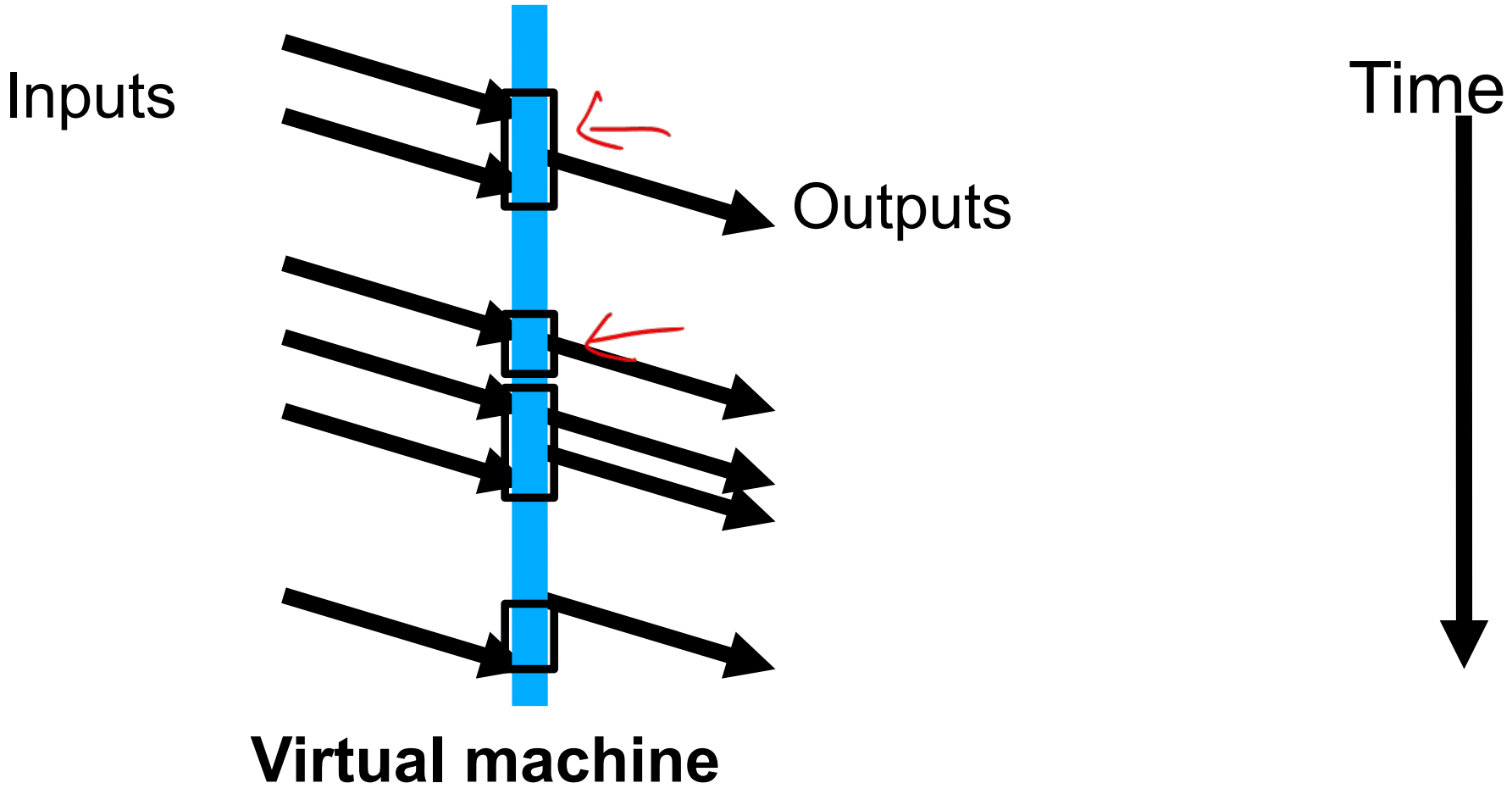


Strawman 2: VM

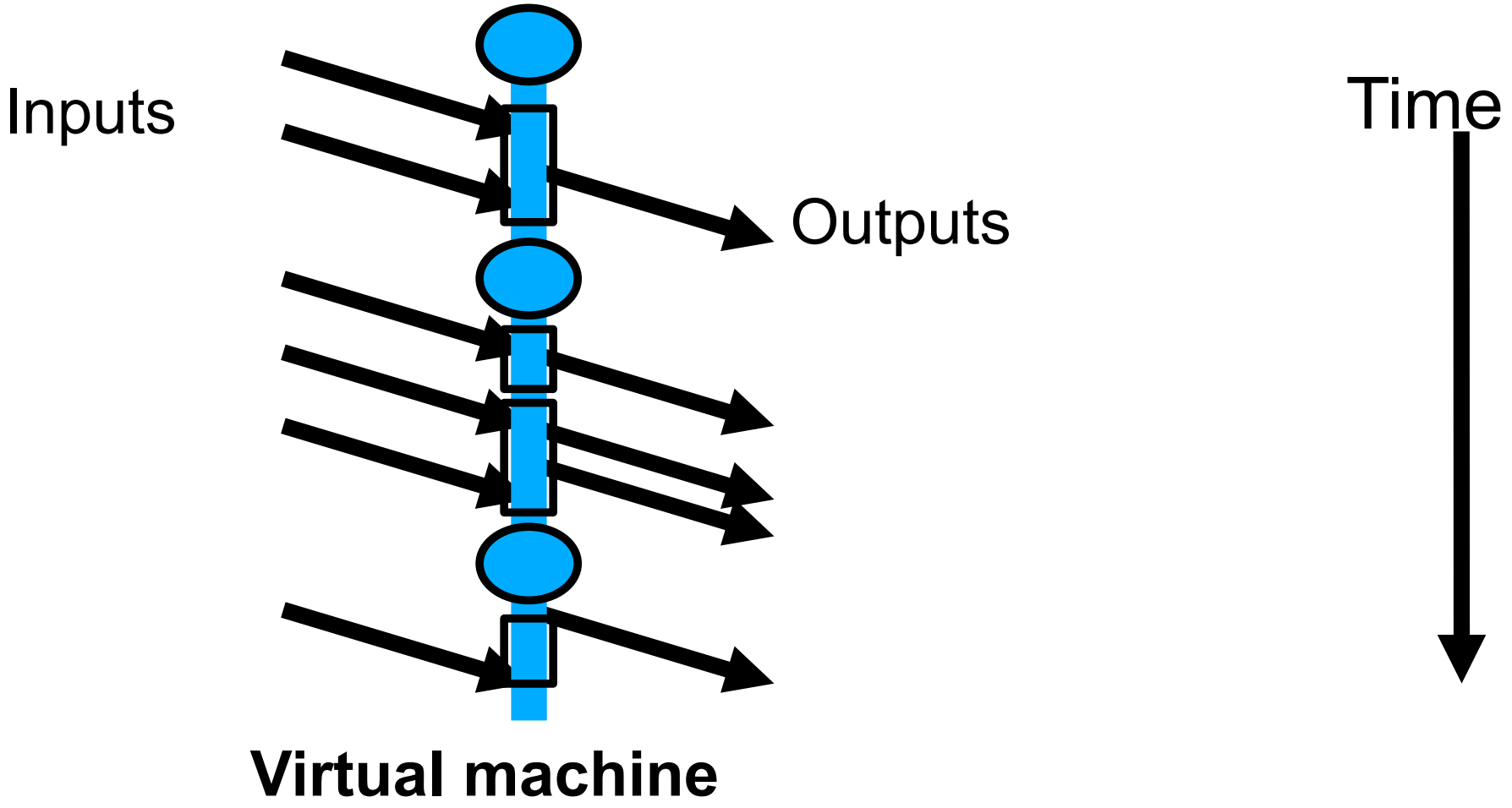
Time

Virtual machine

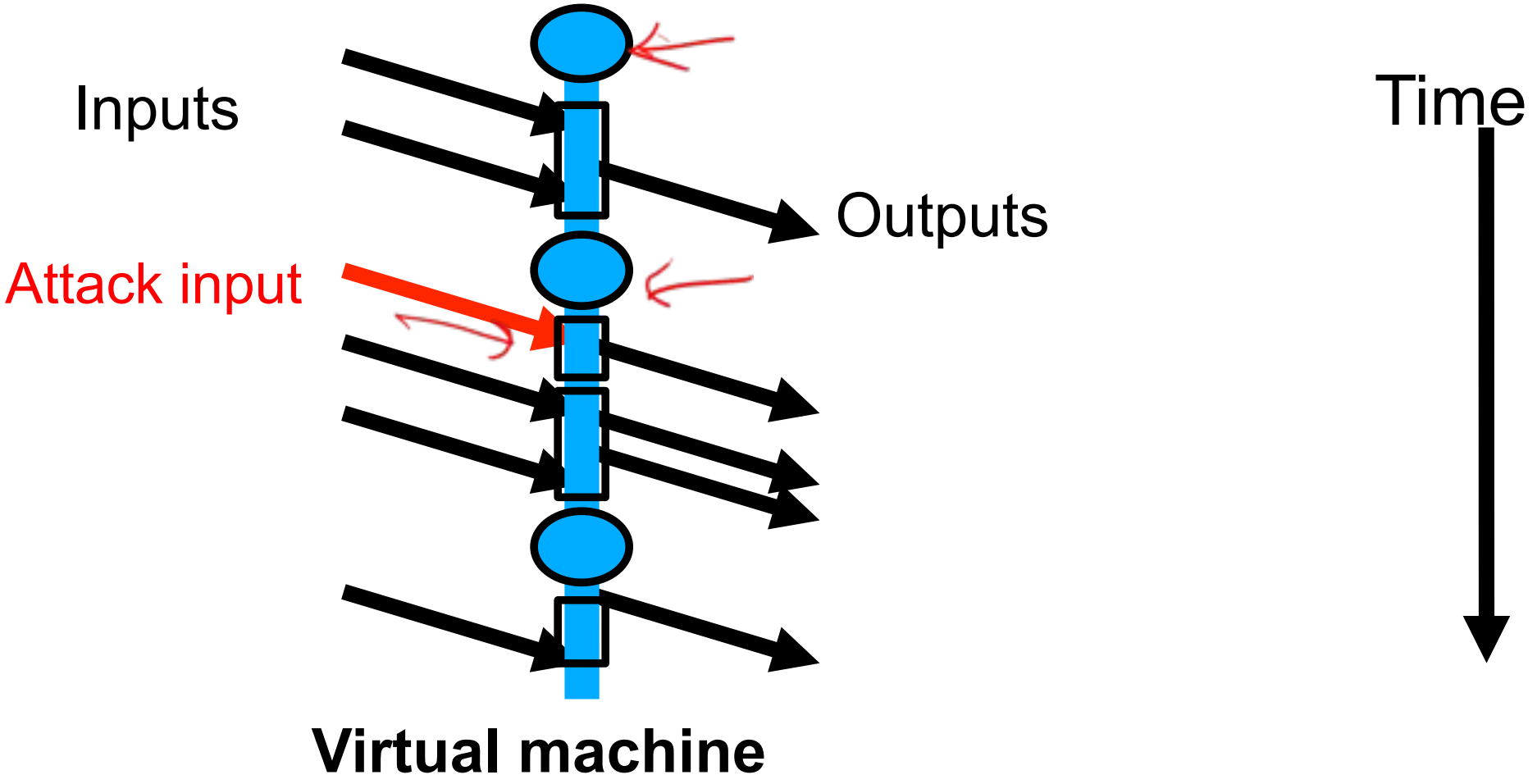
Strawman 2: VM



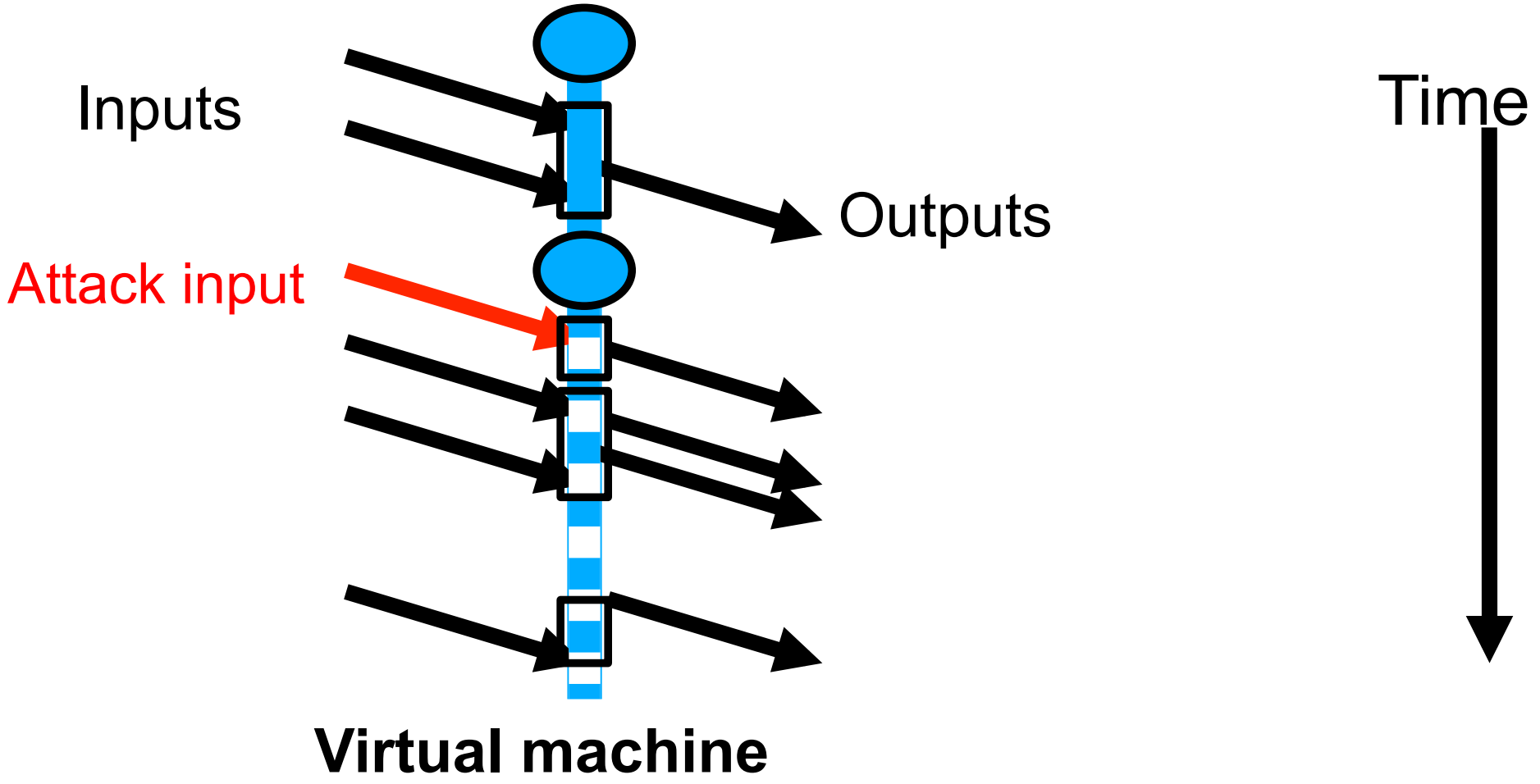
Periodic VM checkpoints



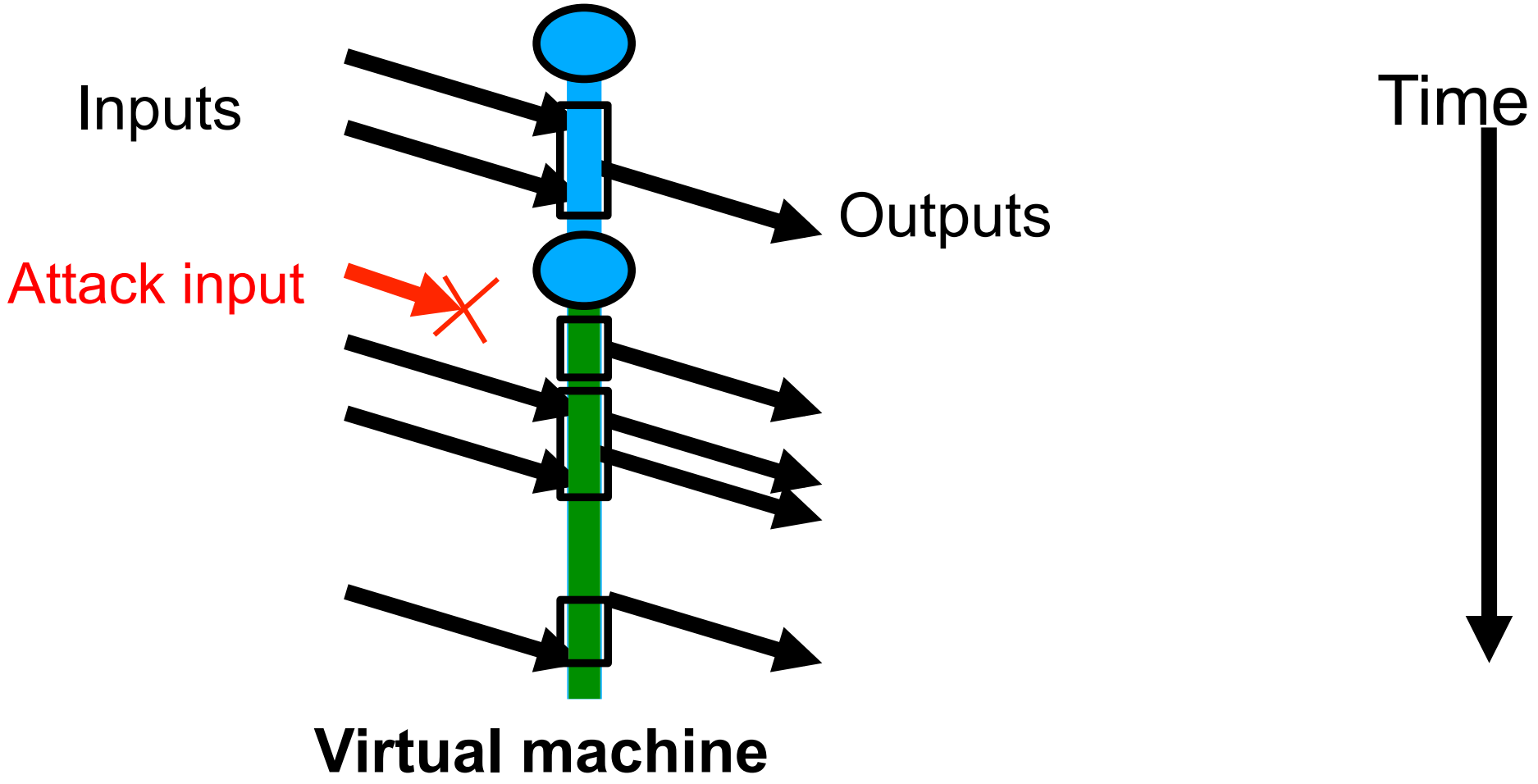
Step 1: identify attack input



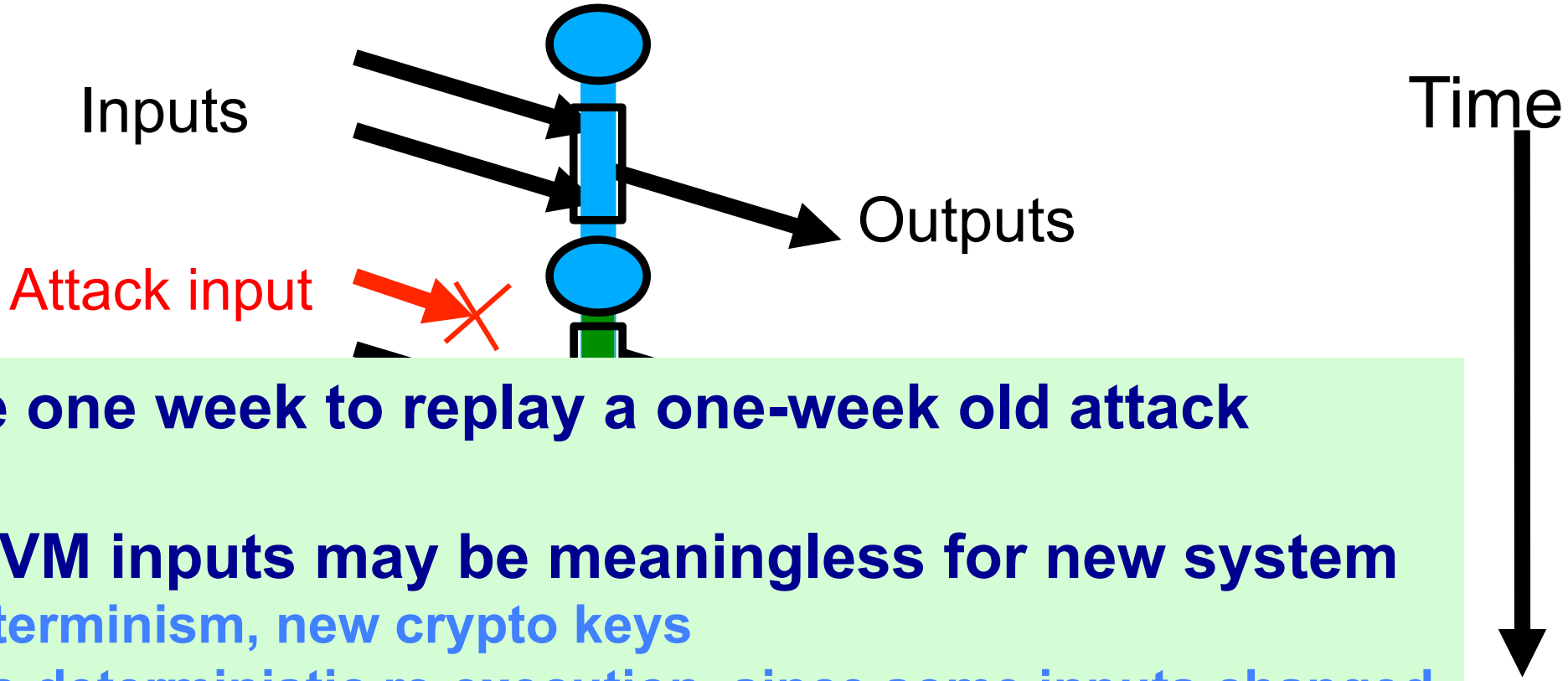
Step 2: roll back to checkpoint



Step 3: replay non-attack inputs



Step 3: replay non-attack inputs



May take one week to replay a one-week old attack

Original VM inputs may be meaningless for new system

- **Non-determinism, new crypto keys**
- **Can't do deterministic re-execution, since some inputs changed**

Retro's approach: selective re-execution

- Record fine-grained *action history graph*
  - Includes system call arguments, function calls, …
  - Assume tamper-proof kernel, storage
- *Roll back* objects directly affected by attack
  - Avoid the false positives of taint tracking
- *Re-execute* actions indirectly affected by attack
  - Avoid expense, non-determinism of whole-VM re-exec.

Summary

- Given just *one* example of each defensive strategy
- Mechanisms corresponding to different strategies have been developed for different layers, e.g., hardware, compiler, operating system
- A secure system may require mechanisms corresponding to *all* three defensive strategies at different layers

THANK YOU

Srini Devadas

Webster Professor of Electrical Engineering and Computer Science

Images used throughout purchased through canva.com