Software Foundations of Security and Privacy (15-316, spring 2016)
Lecture 1: Introduction

Matt Fredrikson

Jean Yang

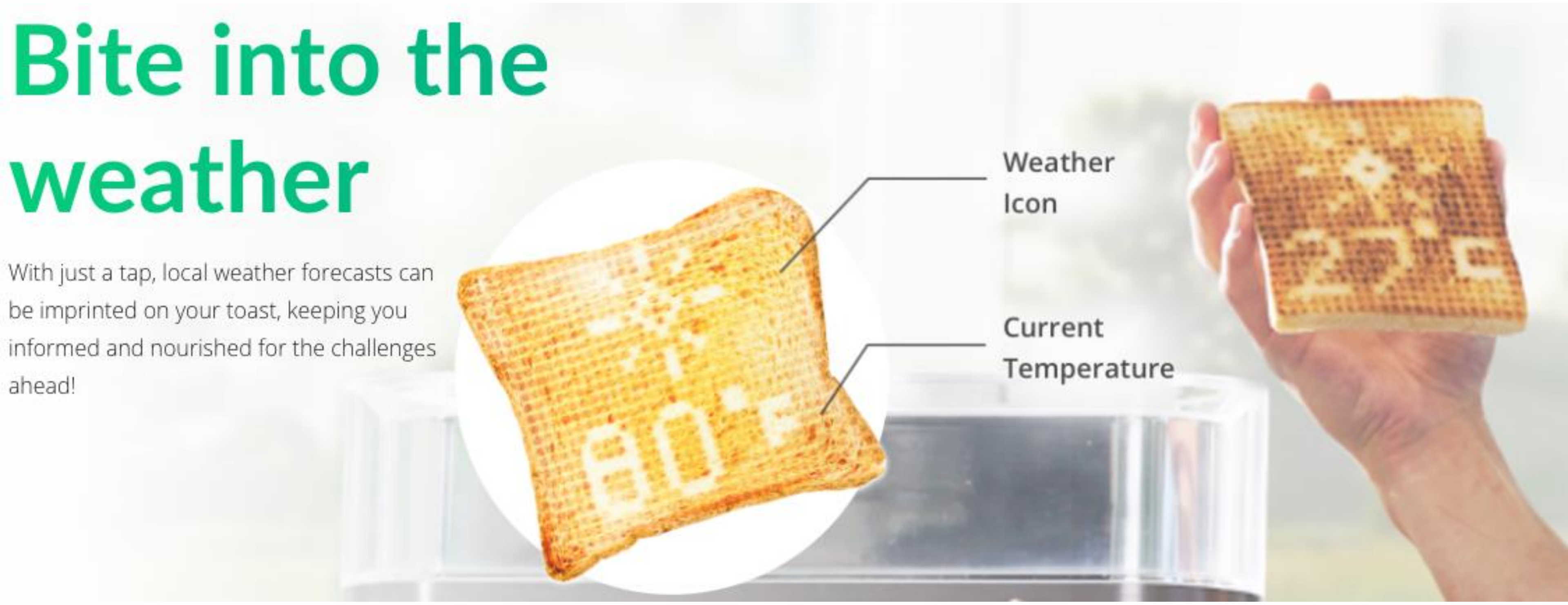
{mfredrik, jyang2}@andrew.cmu.edu

Case study: fancy toasters

WHY POST-IT WHEN YOU CAN TOAST-IT

When sticky notes don't suffice, Toasteroid is here. Never miss an important date or leave your packed lunch behind again. Revolutionary edible reminders brought to you by Toasteroid.

Case study: fancy toasters


Bite into the weather
With just a tap, local weather forecasts can be imprinted on your toast, keeping you informed and nourished for the challenges ahead!
Weather Icon
Current Temperature
80°F
27°C

Existential threats with toasters

- Toast burns and you go hungry.
- Toaster burns and your house burns down.
- Toaster gets hacked, and you neglect to bring your umbrella.
- Someone uses your toasting patterns to figure out when burgle you.
- Someone figures out how to turn on your toaster remotely and your house burns down.

Software threats

Software requirements for toaster

- Toaster should access the internet only to get the weather. What does it mean to be "right?"
- Toaster should get the "right" weather.
- Toaster data should not be available to anyone but "you."
- Toaster should only be operated by "you."

What does it mean to be "you?"

Authentication for smart toasters

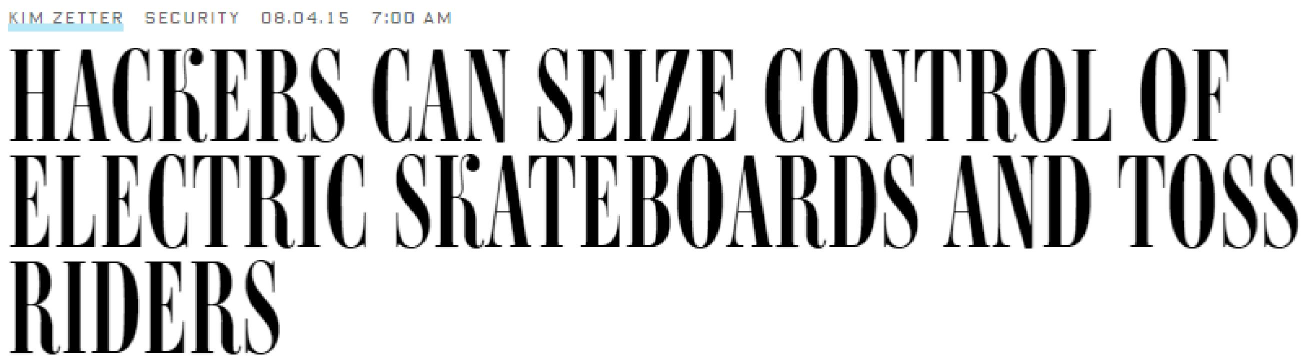
- Most IoT devices controlled by phone apps
- How the toaster establishes trust:
  - Rely on home router to authenticate phone
  - Assume all devices authenticated on the network belong to “you”
- Fine, don’t let any hackers onto your network
  - But, your router knows nothing about your apps
  - Do you trust every app you’ve ever downloaded?

Why we should worry

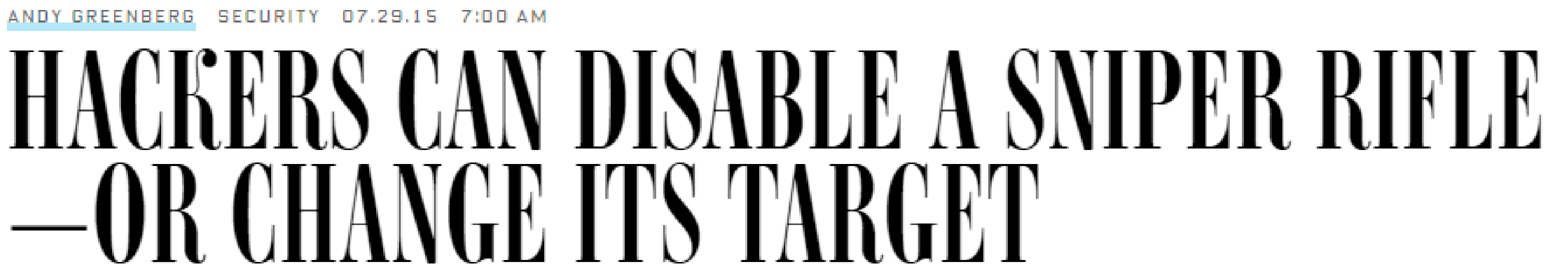
"Assuming it's publicly accessible, the chance [of being hacked] is probably 100 percent… The IPv4 address space just isn't that big. You can now run a scan across that entire space in hours, especially if you have a big botnet. The scans for vulnerability are continuous, and if anything, have accelerated over the last couple of years."

- Matthew Prince, CEO of Cloudflare

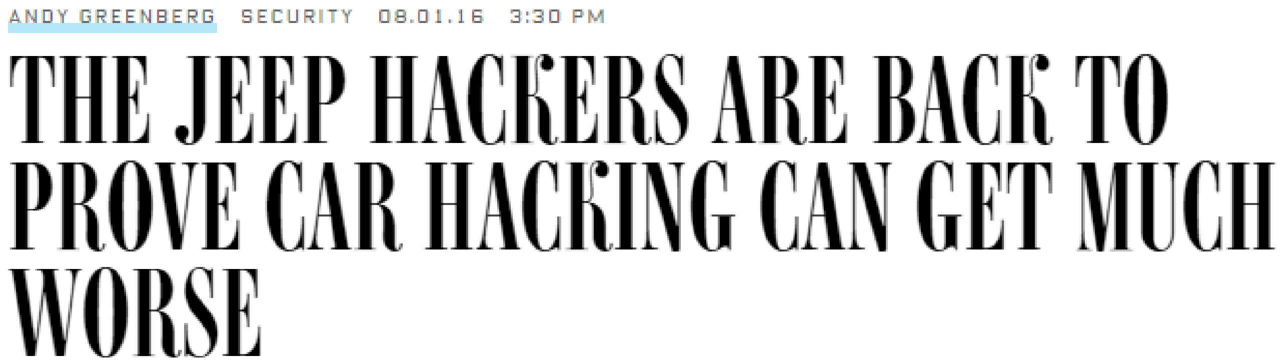
It's not just toasters!


KIM ZETTER SECURITY 08.04.15 7:00 AM

HACKERS CAN SEIZE CONTROL OF ELECTRIC SKATEBOARDS AND TOSS RIDERS

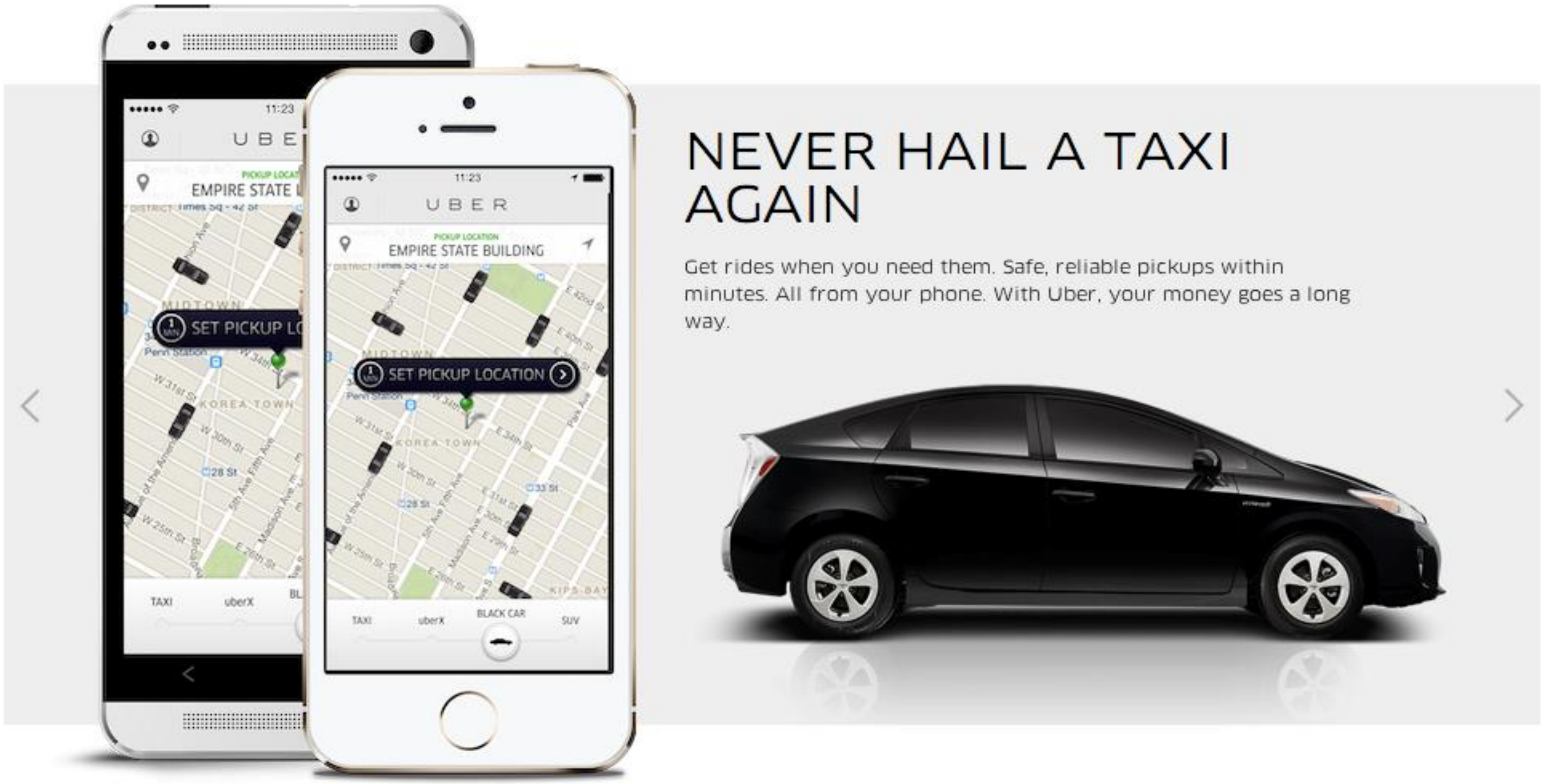

ANDY GREENBERG SECURITY 07.29.15 7:00 AM

HACKERS CAN DISABLE A SNIPER RIFLE —OR CHANGE ITS TARGET


ANDY GREENBERG SECURITY 08.01.16 3:30 PM

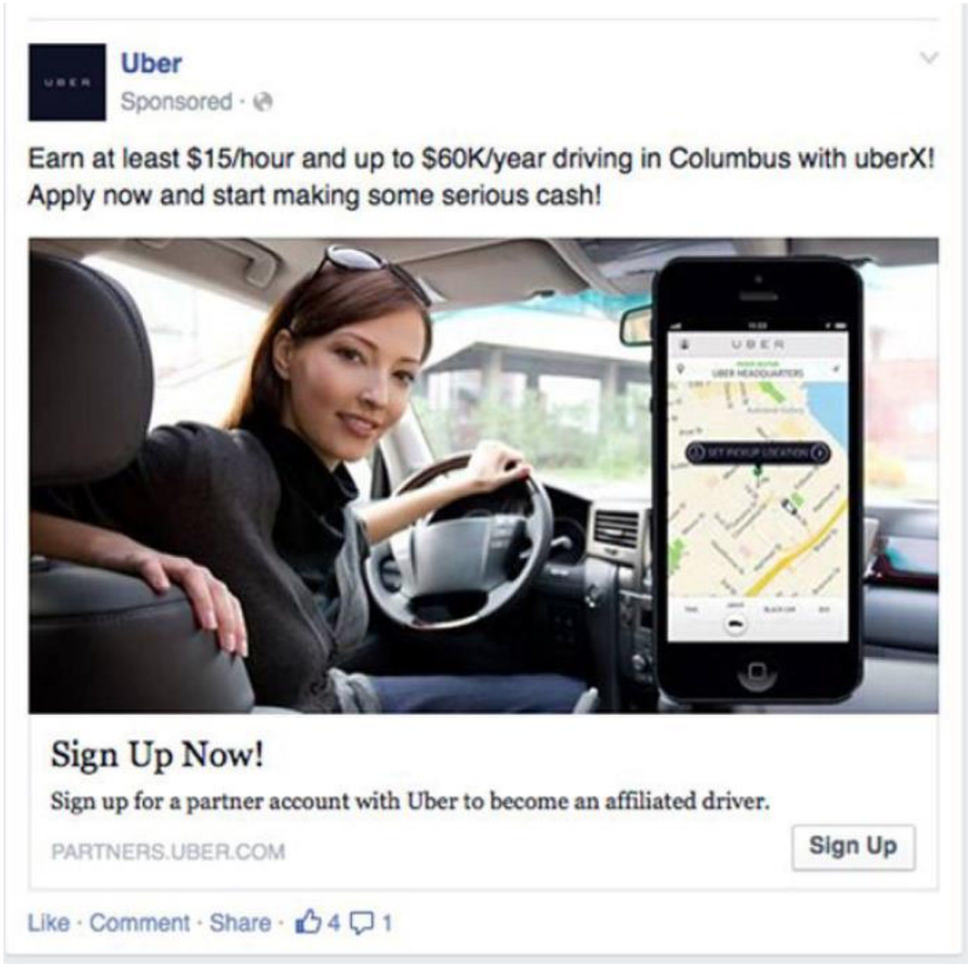
THE JEEP HACKERS ARE BACK TO PROVE CAR HACKING CAN GET MUCH WORSE

Case study: ride-sharing apps



NEVER HAIL A TAXI AGAIN
Get rides when you need them. Safe, reliable pickups within minutes. All from your phone. With Uber, your money goes a long way.
UBER
PICKUP LOCATION
EMPIRE STATE BUILDING
SET PICKUP LOCATION
TAXI
uberX
BLACK CAR
SUV

Case study: ride-sharing apps



Uber
Sponsored ·

Earn at least $15/hour and up to $60K/year driving in Columbus with uberX! Apply now and start making some serious cash!

Sign Up Now!

Sign up for a partner account with Uber to become an affiliated driver.

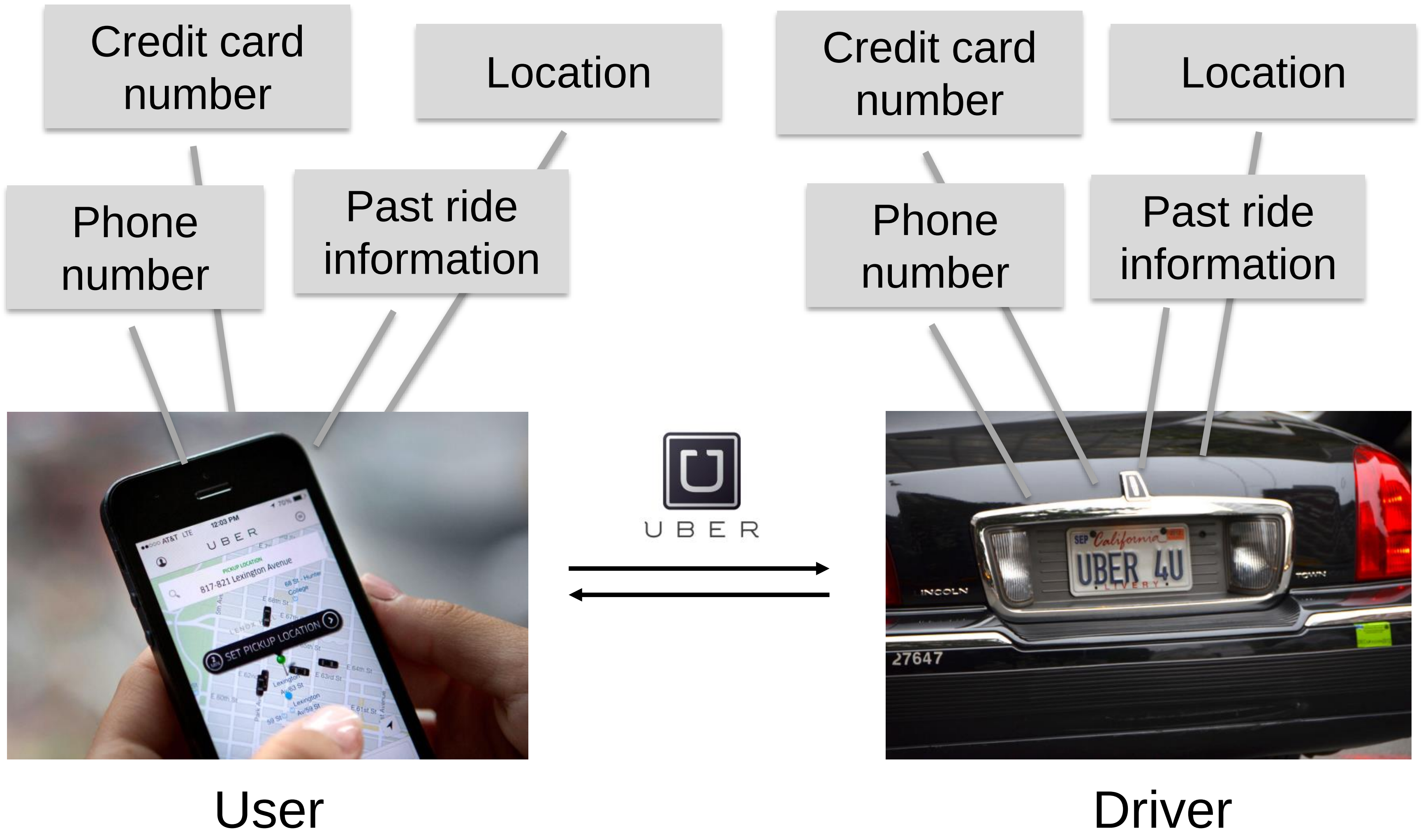
PARTNERS.UBER.COM

Existential threats with Uber

- Driver gets into accident.
- Driver kidnaps us.
- Driver steals our credit card number.
- Our location gets hacked, and we get kidnapped that way.
- People use our locations for stalking and blackmail.

Software threats

How we trust Uber with all data

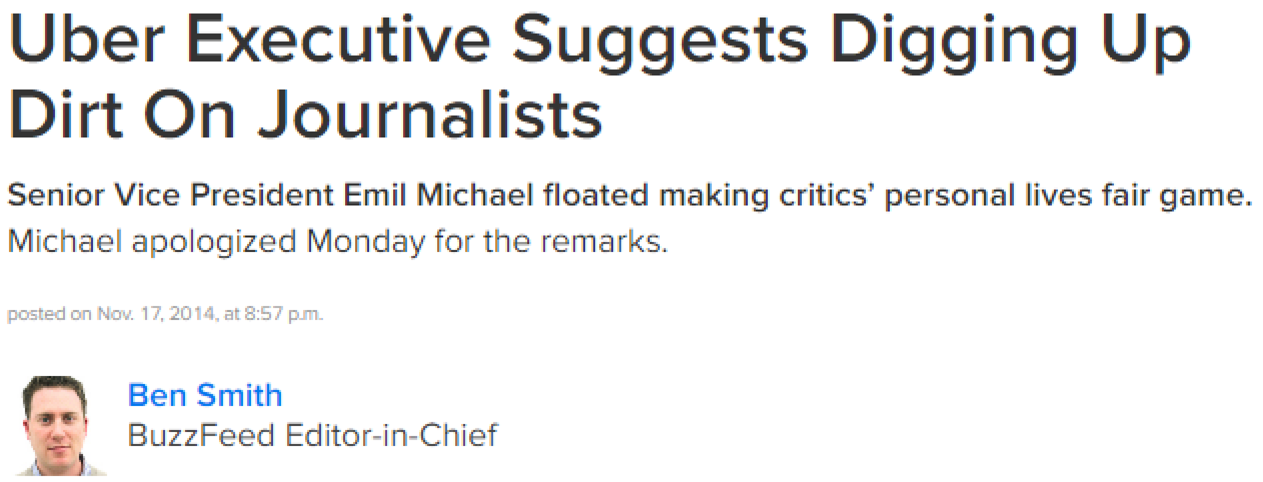


Policies Uber needs to get right

- Only Uber sees credit card numbers.
- Drivers can see rider locations for duration of ride, but not more.
- Riders can see driver locations for duration of ride, but not more.
- Each user can see their own ride history.

Encryption is not enough!

But we can’t even trust Uber!

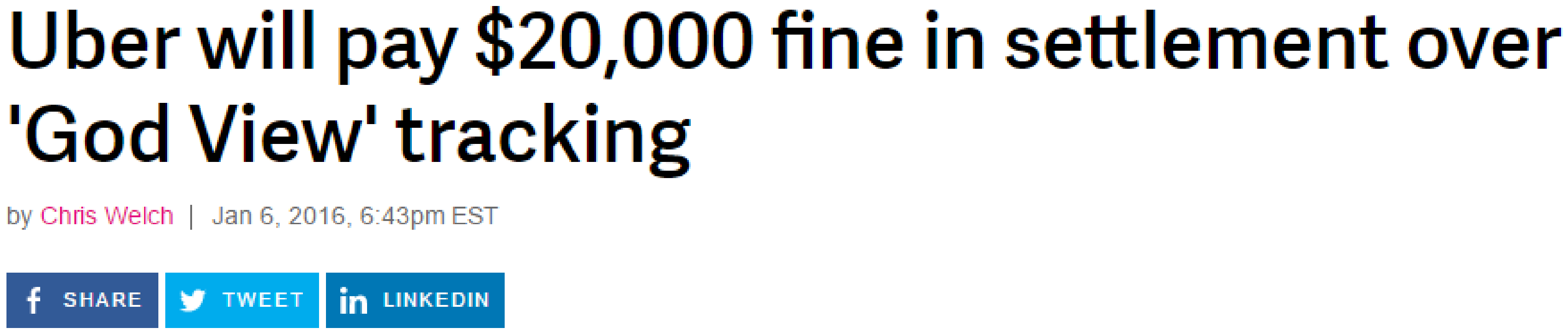


Uber Executive Suggests Digging Up Dirt On Journalists

Senior Vice President Emil Michael floated making critics’ personal lives fair game.
Michael apologized Monday for the remarks.

posted on Nov. 17, 2014, at 8:57 p.m.

Ben Smith
BuzzFeed Editor-in-Chief



Uber will pay $20,000 fine in settlement over 'God View' tracking

by Chris Welch | Jan 6, 2016, 6:43pm EST

SHARE TWEET LINKEDIN

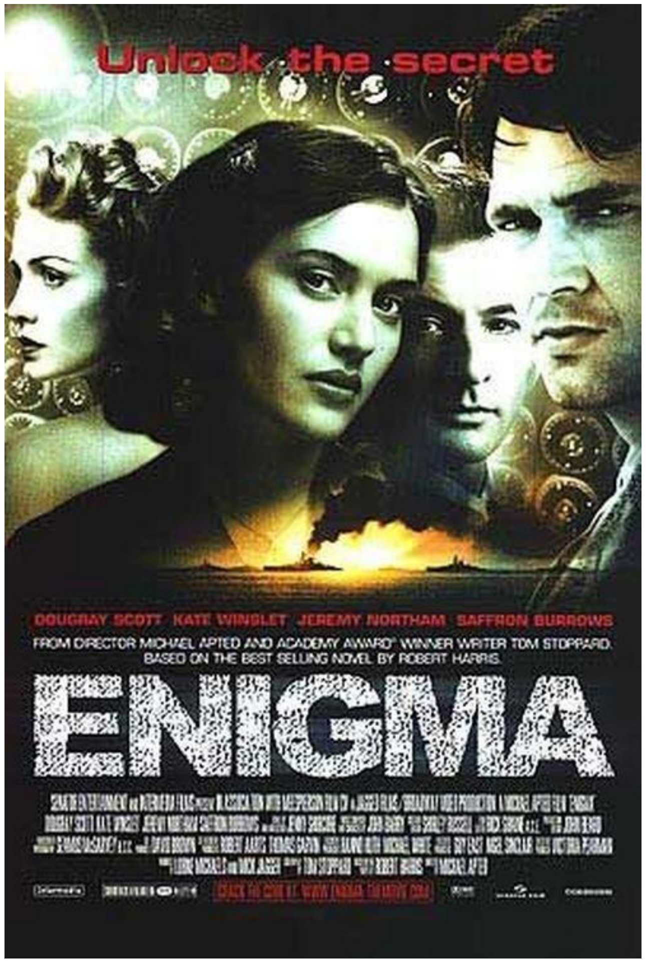
Lessons for the paranoid

- Don’t trust anybody.
- Never, ever use anything connected to the internet.
- Just to be safe, unplug all devices when you are not home.
- Walk everywhere.
- This course can’t help you. Goodbye!

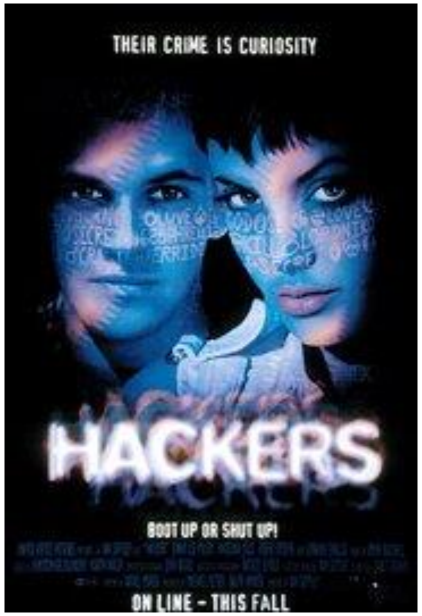
What *Is* Software Security?

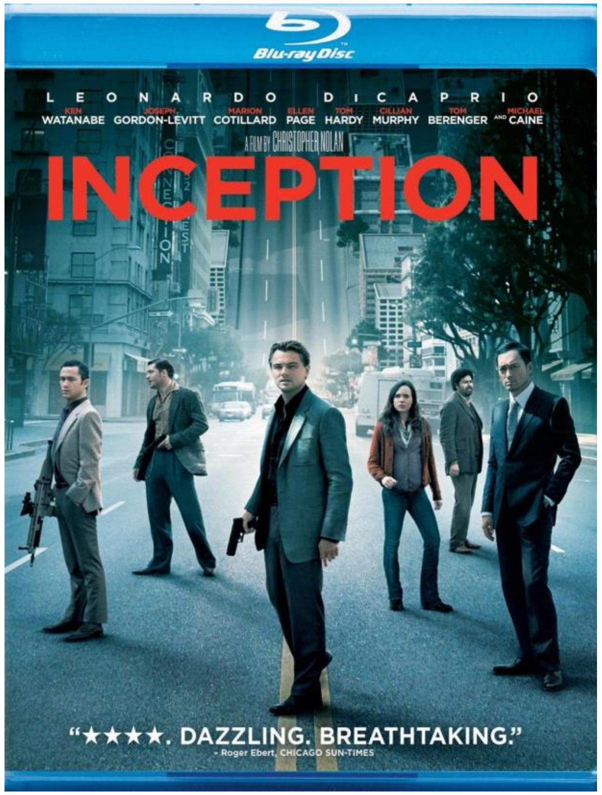
What were you expecting?

This is not a course about encryption


Unlock the secret
DOUGRAY SCOTT KATE WINSLET JEREMY NORTHAM SAFFRON BURROWS
FROM DIRECTOR MICHAEL APTED AND ACADEMY AWARD WINNER WRITER TOM STOPPARD.
BASED ON THE BEST SELLING NOVEL BY ROBERT HARRIS.
ENIGMA

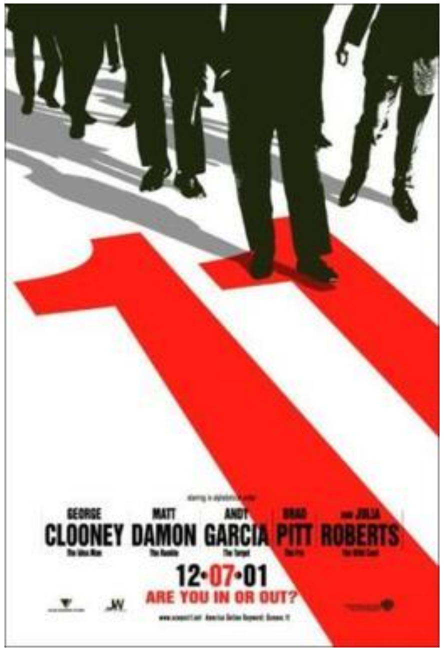
…Not a course about hacking

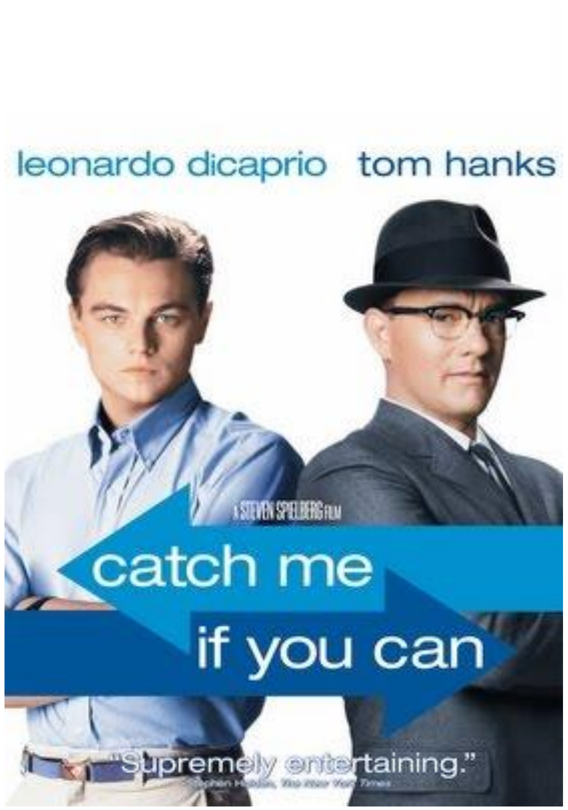




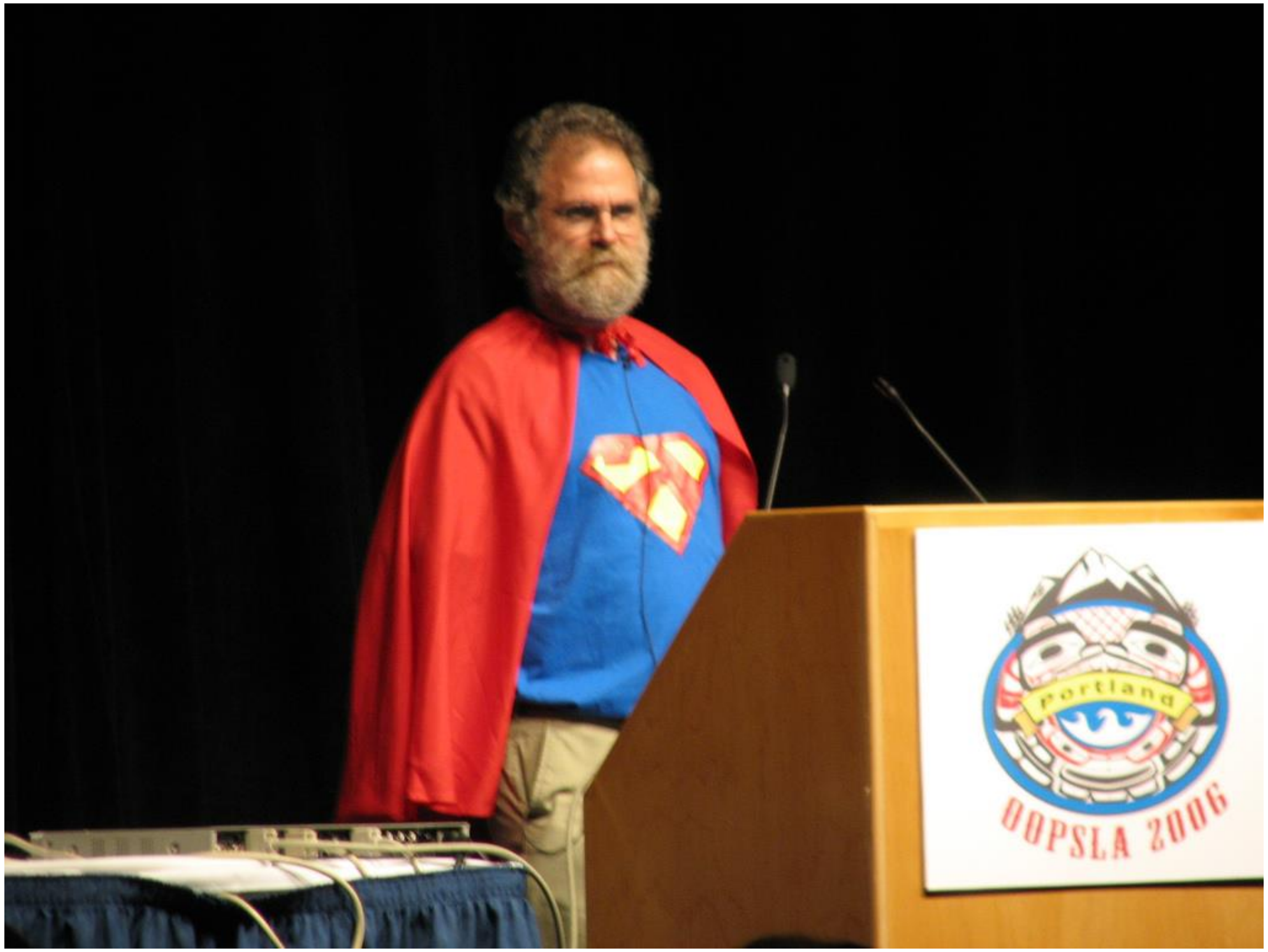
FROM THE INTERNATIONAL BEST-SELLING TRILOGY
DANIEL CRAIG
ROONEY MARA
THE GIRL WITH THE DRAGON TATTOO
COMING SOON

…Not a course about avoiding cons





This is a course about how programming languages will save us



What we ~~want~~ need

- Way of specifying a desired behavior for software.
- Way of specifying who’s allowed to use software/view data, and how.
- Way of ensuring software meets these specifications, **even if we cannot trust the software creators or users**.

What PL and formal methods give us

- Ways of formally stating specifications about program behavior.
- Ways of formally stating specifications about security and privacy.
- Ways of verifying and testing programs with respect to specification.
- Mechanisms for auditing third-party programs with sound guarantees.

What being formal gives us

- We’re after **formal security guarantees**

- A formal guarantee is:
  - *Precise and unambiguous*: no surprises, all of our assumptions are made explicit
  - *Provable/refutable*: able to determine whether it holds using logical reasoning
  - Ideally, amenable to *mechanized, automated* reasoning: less work for us, and much less prone to human error

What being formal doesn't give us

- At the end of the day, systems run in the "real world"
- Formalism isn't a panacea
  - Proofs are always relative to the *formal definition* and any *assumptions* in play
  - When these aren't realistic, the guarantee isn't meaningful in practice
  - Still need creativity and good engineering to devise useful definitions and implementations

What this course is about

- How can we reason about what software is doing?
  - How do we specify intended behavior of both the program and of data release?
  - How do we get guarantees about behavior?
- How do we build systems in ways that **provably protect** security and privacy?

Principles of the course

- Correctness is the foundation of security.
  - Make programmer assumptions as explicit as possible.
  - Make abstraction boundaries as airtight as possible.
- The easier it is to reason about what software is doing, the more secure the software will be.

Security vs. privacy

Security

Privacy

Security vs. privacy: one definition

- **Security** is about making sure the goals of a system can be achieved regardless of the presence of an adversary.
- **Privacy** is about the rights of the user and their data, and protecting these from unwanted exposure.

Questions and comments?

Basic Principles of Secure Software

Least privilege

Every program and every user should operate using the least set of privileges necessary to complete the job.

- Example: military “need-to-know.”
- Anti-example: Uber’s “God mode.”

Smallest trusted computing base

- The **trusted computing base** (TCB) is a subset of the system essential to its security.
  - Kernel, network stack, hardware platform, …
  - Exact “boundary” depends on policy
- Why do we want a small TCB?
  - Less code $\rightarrow$ less room for error $\rightarrow$ less risk
  - Need comprehensive tests, or detailed proofs
  - Rigorous validation is expensive, time-consuming

Complete mediation

Every access to every object must be checked for authority.

- Is it enough to check at the application endpoints? (HotCRP email password reminder bug)
- What should the policy be? (average salary example)

In this course…

- We will see how to apply these principles to **large, complex software systems.**
- We will see how to apply these principles with **complex, state-dependent policies.**
- We will **build our own systems** that use this principles—and attack them!

Questions and comments?

Course Logistics

What we expect you to know

- Official prerequisites: 15-122, 15-213.
- How to teach yourself a programming language.
- How to build software systems.
- How to reason formally about computation.

Unit Overview

Unit	Theme
One	Single-user program correctness, the foundation of security and privacy.
Two	What happens when we have multiple users?
Three	How can we track sensitive values across complex programs?
Four	From security to privacy: statistical privacy.

Example topics we will cover

- Formally stating and proving correctness properties with automata.
- Techniques for verifiable security assurance in realistic systems.
- Mechanisms for authentication and identity.
- Language-based information flow.
- Differential privacy.

Background: Build It Break It Fix It

- Main questions
  - What goes into secure software development?
  - How do we teach people to write secure code?
- Contest   Focus of our course assignments
  - **Build** software according to specifications
  - **Break** software written by other contestants
  - **Fix** bugs found in their software
- Tasks are doable over 72 hours

Project Overview

- Implement an "enhanced" version of the Build It Break It Fix It server
- Progressively improve functionality alongside security
- Some what you will build:
  1. Basic data server with limited policy support
  2. Reference monitor to support flexible, user-defined data policies
  3. *Proof-carrying authorization* for fine-grained access policies
  4. *Information flow types* to control data leakage
  5. *Differential privacy* policies to protect individuals' data

Grading

- 60% Assignments
- 30% Midterm and final exams
- 10% In-class quizzes and participation

What You Will Learn

Specifics

- How to build a data server, and implement mechanisms to secure it in complementary ways.
- How to formally reason about security and privacy properties of this server.
- How to prove pencil-and-paper properties about your server.
- How to attack your server and how to code in a way that defends against attacks.

Ways of thinking

- How to formally specify software correctness.
- How to formally reason about security goals and threat models.
- How to build systems that are secure by construction.
- How to think rigorously about attacks.

What now?

1. Make sure you sign in so we can add you to Piazza.
2. Bookmark our course website (http://cs.cmu.edu/~15316).
3. Do the OCaml finger exercises (from the course website).
4. Email us the necessary items from Assignments 0, and talk to us if you have questions about the course!