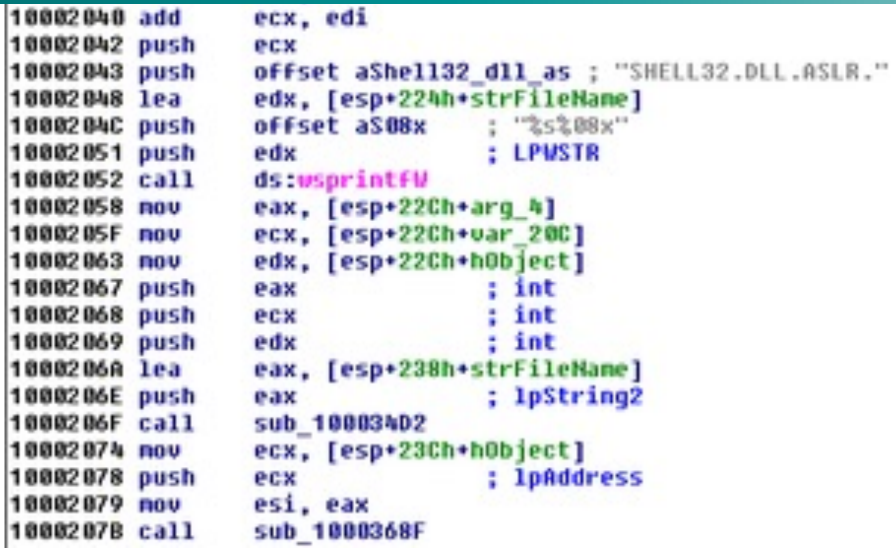
Lecture 1: Introduction to Security

CS 181W

Fall 2022

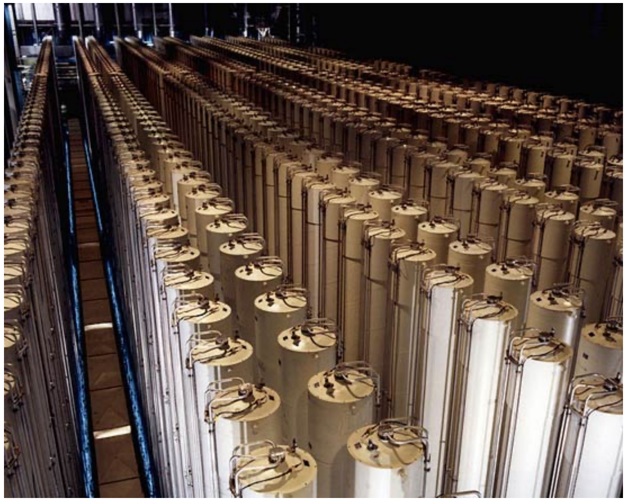
```
static report_breakin(arg1, arg2)                  /* 0x2494 */
{
    int s;
    struct sockaddr_in sin;
    char msg;

    if (7 != random() % 15)
          return;

    bzero(&sin, sizeof(sin));
    sin.sin_family = AF_INET;
    sin.sin_port = REPORT_PORT;
    sin.sin_addr.s_addr = inet_addr(XS("128.32.137.13"));
```

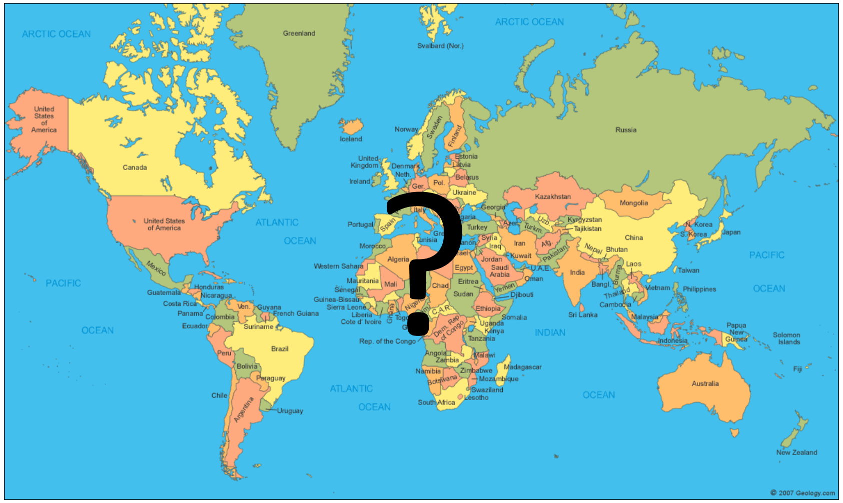
November 2, 1988



```
10002040 add     ecx, edi
10002042 push    ecx
10002043 push    offset aShell32_dll_as ; "SHELL32.DLL.ASLR."
10002048 lea     edx, [esp+224h+strFileName]
1000204C push    offset aS08x     ; "%s%08x"
10002051 push    edx              ; LPWSTR
10002052 call    ds:wsprintfW
10002058 mov     eax, [esp+22Ch+arg_4]
1000205F mov     ecx, [esp+22Ch+var_20C]
10002063 mov     edx, [esp+22Ch+hObject]
10002067 push    eax              ; int
10002068 push    ecx              ; int
10002069 push    edx              ; int
1000206A lea     eax, [esp+238h+strFileName]
1000206E push    eax              ; lpString2
1000206F call    sub_100034D2
10002074 mov     ecx, [esp+23Ch+hObject]
10002078 push    ecx              ; lpAddress
10002079 mov     esi, eax
1000207B call    sub_1000368F
```

June 1, 2012





August 25, 2022

okta

Signal

DOORDASH

INTERESTING

HARD

Today

FUN

IMPORTANT

Defining security

“This tops the list of recommendations for upgrading your online security.”

Functional Requirements

- **Security** = **does what it should** + nothing more
- "As a *user* I can *action* so that *purpose*"
  - e.g., As a professor, I can create a new assignment by specifying its name, number of possible points, and due date.
  - e.g., As a student, I can upload a file as a solution to an assignment.
  - e.g., As a professor, I can assign grades to student solutions.

Functional requirements should specify **what** not **how**

- Should be **testable**: a 3rd party could determine whether requirement is met
- These user stories reveal system **assets**

Security Goals

- **Security** = does what it should + **nothing more**
- "The system shall prevent/detect *action* on/to/with *asset*."
  - e.g., "The system shall prevent students from accessing assignments that are not theirs"
  - e.g., "The system shall prevent grades from being changed by anyone but the professor"

Security goals should specify **what** not **how**

- Poor goals:
  - "the system shall use encryption to prevent reading of messages"
  - "the system shall use authentication to verify user identities"
  - "the system shall resist attacks"
- If a system enforces a goal, it is called a **security property**

C I A

CIA

Confidentiality

Integrity

Availability

Confidentiality Properties

Protection of assets from unauthorized disclosure

i.e., which principals are allowed to learn what

Examples:

- Keep contents of a file from being read (*access control*: more later)
- Keep information secret (*information flow*: more later)
  - value of variable secret
  - behavior of system
  - information about individual

Integrity Properties

Protection of assets from unauthorized modification

i.e., what changes are allowed to system and its environment, including inputs and outputs

Examples:

- Output is correct according to (mathematical) specification
- No exceptions thrown
- Only certain principals may write to a file (access control)
- Data are not corrupted or tainted by downloaded programs (information flow)

Availability Properties

Protection of assets from loss of use

i.e., what has to happen when/where

Examples:

- Operating system accepts inputs periodically
- Program produces output by specified time
- Requests are processed fairly (order, priority, etc.)

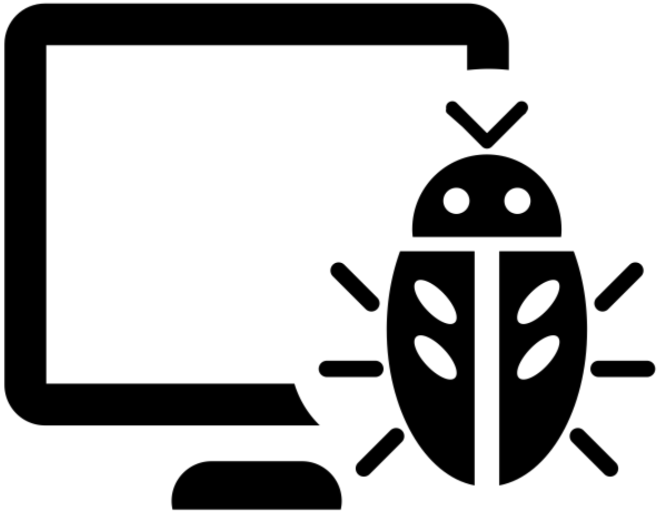
Denial of service (DoS) attacks compromise availability

A Secure Grade Management System

1. Students can always log into their accounts
2. The grade for an assignment is available only to the student who submitted that assignment.
3. The professor can see all submitted assignments and grades.
4. If your course grade changed, then the professor made that change.
5. If your course grade changed, you see the updated grade.
6. Requests to the grading server are processed in the order they were received.

Attackers exploit bugs

- Software bugs
- Hardware bugs
- Humans (social engineering)
- Unintended characteristics
  - side channels
  - poor sources of randomness
  - …



Created by iconoci
from Noun Project

Modeling the attacker

- What type of action will they take?
    - Passive (look, but don’t touch)
    - Active (look and inject messages)
- How much do they already know?
    - External / internal attacker?
- How sophisticated are they?
- How much do they care? What resources do they have?
    - How much time/money will they spend?

Created by Jorge Reyes
from the Noun Project

Exploiting bugs as a nuisance

- Pranks, to be annoying
  - Newsday tech writer & hacker critic found …
    - Email box jammed with thousands of messages
    - Phone reprogrammed to an out of state number where caller's heard an obscenity-loaded recorded message [Time Magazine, December 12, 1994]
- May be costly
  - MyDoom (2004) - $38.5 billon
  - SoBig (2003) - $37.1 billion
  - Love Bug (2000) - $15 billion
  - Code Red (2001) - $2 billion

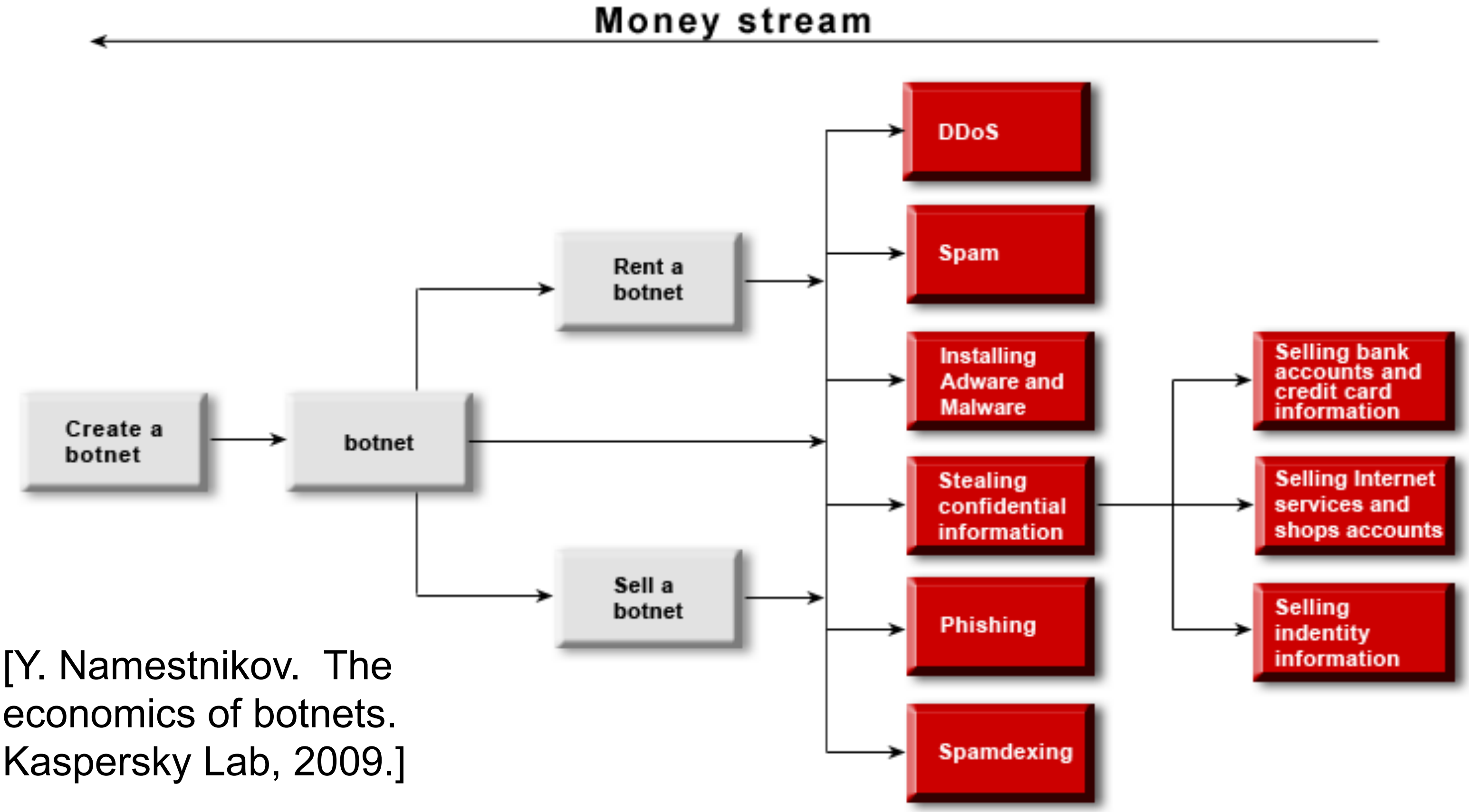
Created by Michael Thompson
from Noun Project

Exploiting bugs for profit

- Credit card and financial account fraud
- Stealing intellectual property or confidential information
- Ransom
- Extortion
- Stealing computing resources to sell

Created by Gregor Cresnar
from Noun Project

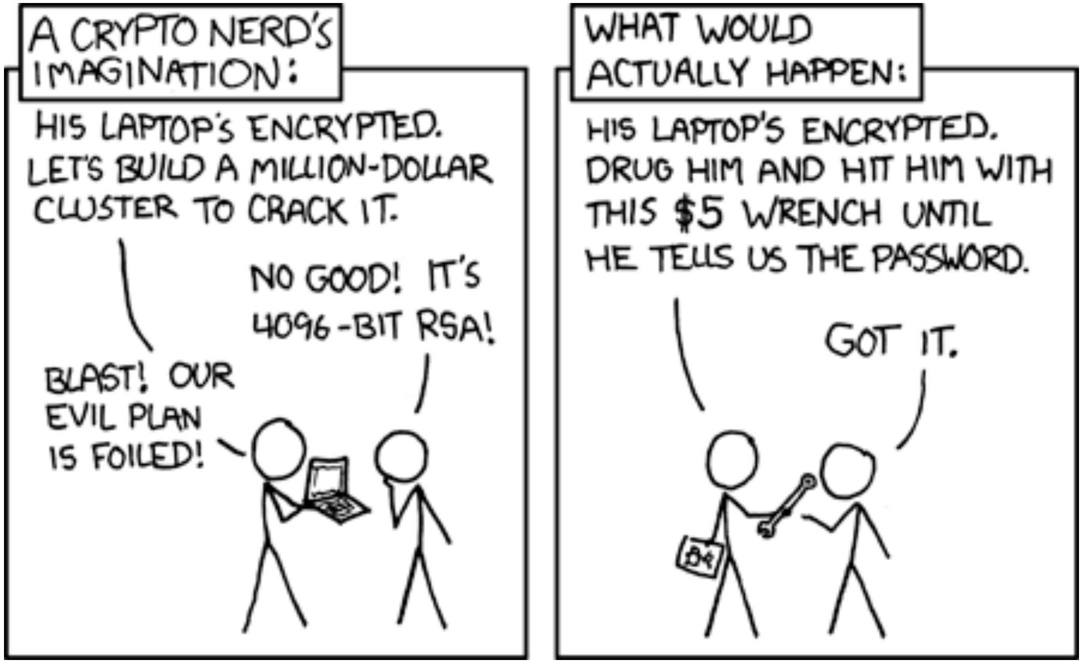
The economics of botnets


Money stream
Create a botnet
botnet
Rent a botnet
Sell a botnet
DDoS
Spam
Installing Adware and Malware
Stealing confidential information
Phishing
Spamdexing
Selling bank accounts and credit card information
Selling Internet services and shops accounts
Selling indentity information

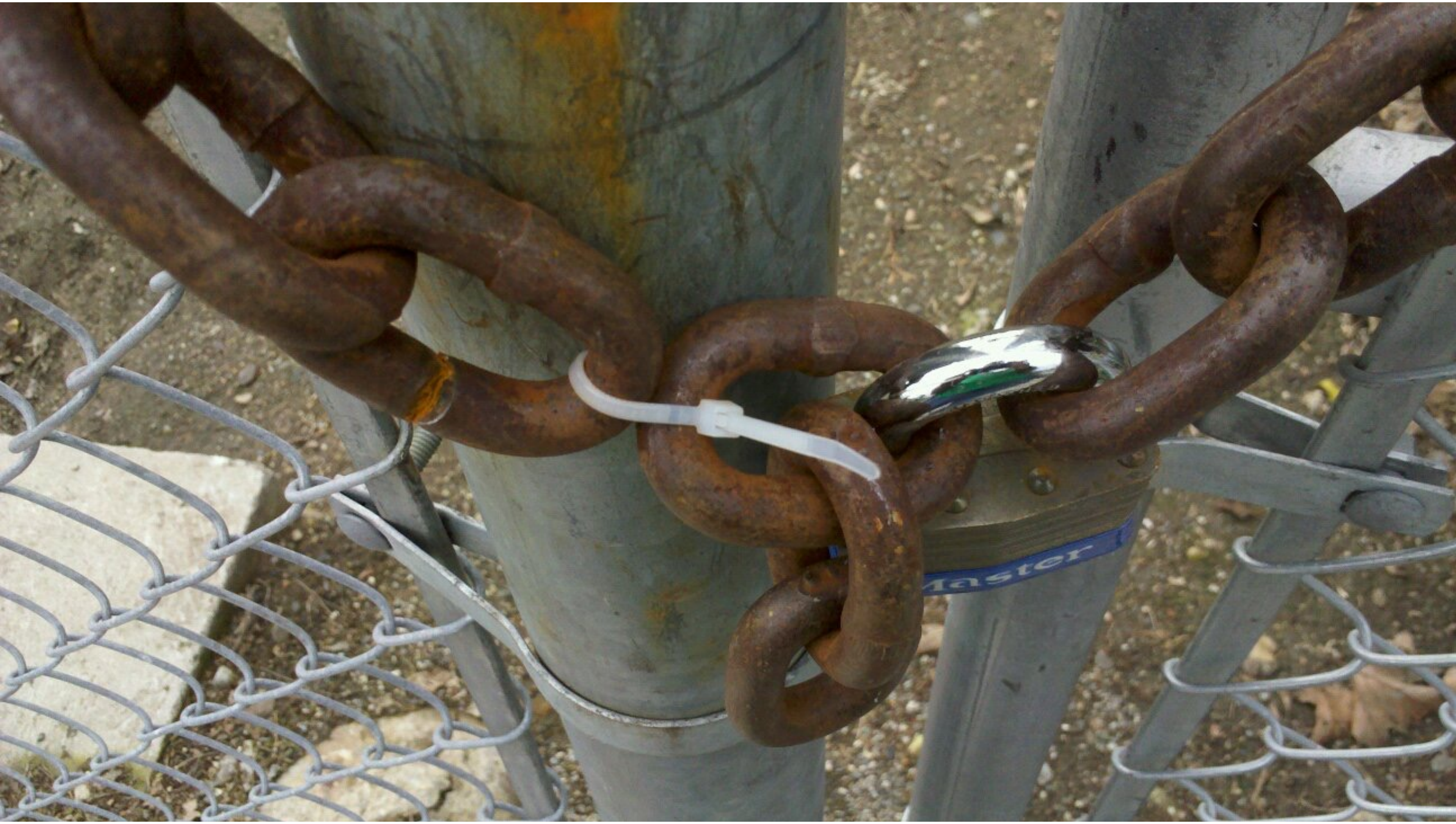
[Y. Namestnikov. The economics of botnets. Kaspersky Lab, 2009.]

Think like an attacker

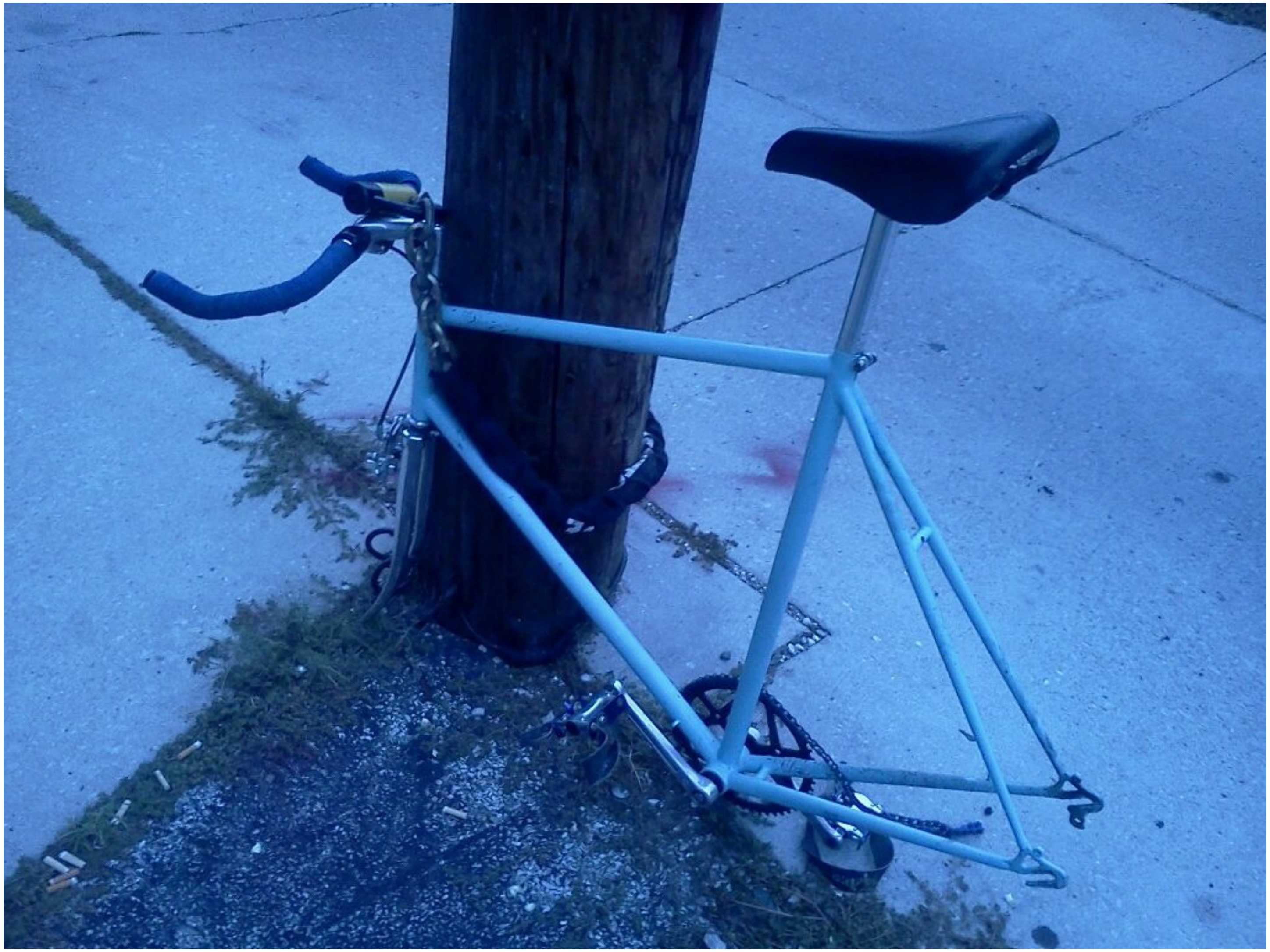
- Adversary is targeting *assets,* not defenses
- Will try to exploit the *weakest* part of the defenses
  - E.g., bribe human operator, social engineering, steal (physically) server with data


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

What will be attacked?



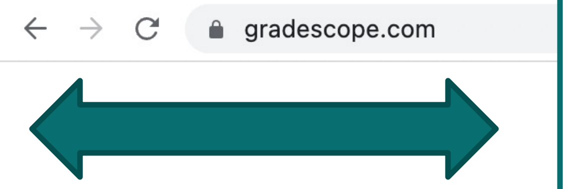
What was being defended?

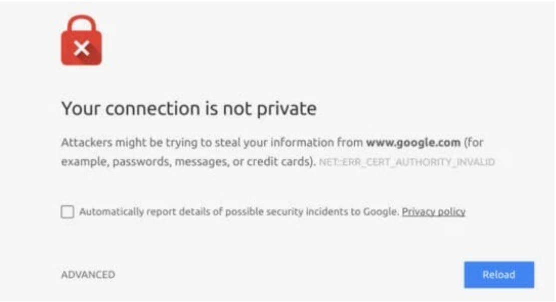


A Secure Grade Management System

1. Students can always log into their accounts
2. The grade for an assignment is available only to the student who submitted that assignment.

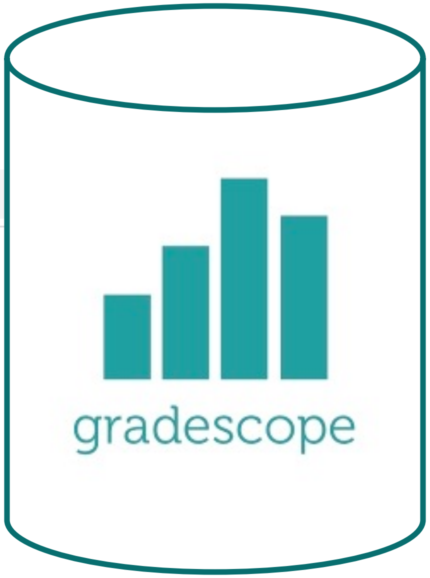
Name
Your Name
Email Address
email@example.com
Student ID
Sign up as a student
DUO




Your connection is not private

Attackers might be trying to steal your information from **www.google.com** (for example, passwords, messages, or credit cards). NET::ERR_CERT_AUTHORITY_INVALID

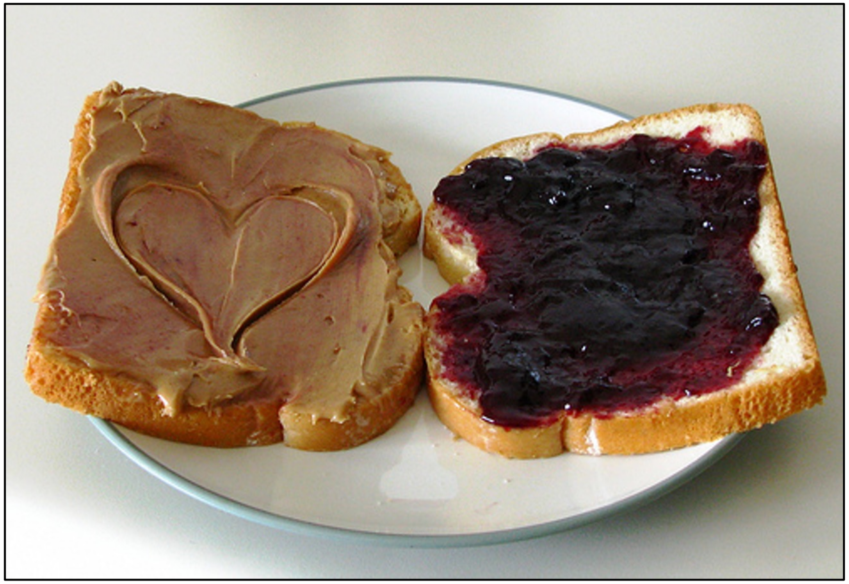
□ Automatically report details of possible security incidents to Google. Privacy policy



HUMANS
ARE THE
PROBLEM

Better together

Examining security/privacy and usability together is often critical for achieving either



Interdisciplinary approach useful

We can borrow models and methods from other disciplines that study human behavior:

- HCI
- Psychology
- Sociology
- Ethnography
- Cognitive sciences
- Behavioral economics
- Warnings science
- Risk perception
- Organizational change
- Marketing
- Counterterrorism
- Communication
- Persuasive technology
- Learning science
- Legal theory

What makes usable security different?

- Presence of an adversary or risk
- Usability is not enough
- We also need systems that remain secure when:
  - Attackers (try to) fool users
  - Users behave in predictable ways
  - Users are acting under stress
  - Users are careless, unmotivated, busy

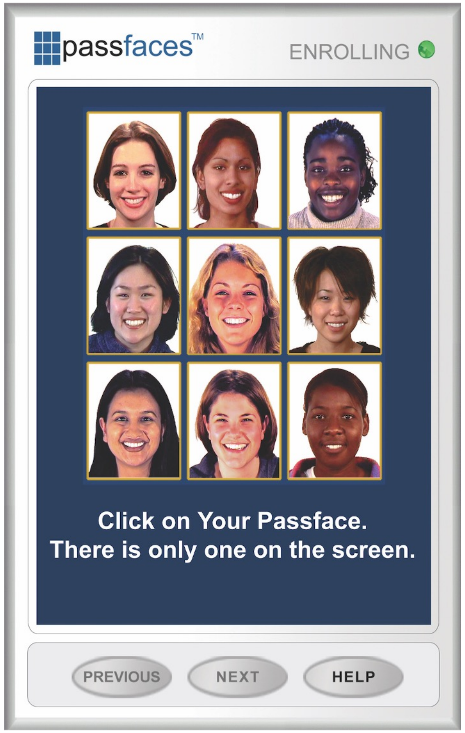
Security and usability together

Security	Usability/HCI	Usable Security
Humans are a secondary constraint to security constraints	Humans are the primary constraint, security rarely considered	Human factors and security are both primary constraints

Security and usability together

Security	Usability/HCI	Usable Security
Humans are a secondary constraint to security constraints	Humans are the primary constraint, security rarely considered	Human factors and security are both primary constraints
Humans considered primarily in their role as adversaries/attackers	Concerned about human error but not human attackers	Concerned about both normal users and adversaries
Involves threat models	Involves task models, mental models, cognitive models	Involves threat models AND task models, mental models, etc.
Focus on security metrics	Focus on usability metrics	Considers usability and security metrics together
User studies rarely done	User studies common	User studies common, often involve deception + active adversary

Example: graphical passwords



Click on Your Passface.
There is only one on the screen.

Example: graphical passwords

Security	Usability/HCI	Usable Security
What is the space of possible passwords? How can we make the password space larger to make the password harder to guess? How are the stored passwords secured? Can an **attacker** gain knowledge by observing a user entering her password?	How *difficult* is it for a **user** to create, remember, and enter a graphical password? How long does it take? How hard is it for users to learn the system? Are users *motivated* to put in effort to create good passwords? Is the system *accessible* using a variety of devices, for users with disabilities?	All the security/privacy and usability HCI questions How do **users** select graphical passwords? How can we help them choose passwords harder for **attackers** to predict? As the password space increases, what are the impacts on usability factors and predictability of human selection?

LOGISTICS

Course Logistics

Prof. Eleanor Birrell

Research in usable security and privacy
OH: T 4-6pm PT + TBA

- **Class Meetings:**
  - Monday and Wednesday, 11:00am-12:15pm PT in Lincoln 1135

Course Work

- Final course project (35%)
  - Conduct an experiment in usable security or usable privacy
  - Done in groups of 3-4
- Homework Assignments (40%)
  - Approximately 8 assignments
  - Mostly building towards your final project
- Reading Assignments (20%)
  - Read papers and write brief summaries
- Participation (5%)
  - Show up! Participate! Have fun!

- All assignments will be due Tuesdays at 11:59pm PT

Course website

https://cs.pomona.edu/~ebirrell/classes/cs181w/2022fa/

- All information is on the course website

CS 181W: Usable Security and Privacy

Copyright © Randy Glasbergen. www.glasbergen.com