Introduction to Cybersecurity and Security Mindset

Shuai Wang

香 港 科 技 大 學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Agenda

- Motivation & definition about “Security”
- Some interesting topics in security research
- Mindset

Learn to become a “hacker”, an ethic one ☺

Cybersecurity is Real-World Problem-Driven

- Many (research) topics are indeed driven by security breaches in the real world!
  - That's one key reason I decide to work in this field

A Sad (?) Story

aws

Amazon Web Services

Mon May 18 2020 10:51:26

Translate

Was this response helpful? Click here to rate:

★★★★★

Hi there,

wangshuai

Hello,

aws

Amazon Web Services

Fri Jun 05 2020 15:00:49 GMT+0800 (Hong Kong Standard Time)

Translate

Was this response helpful? Click here to rate:

★★★★★

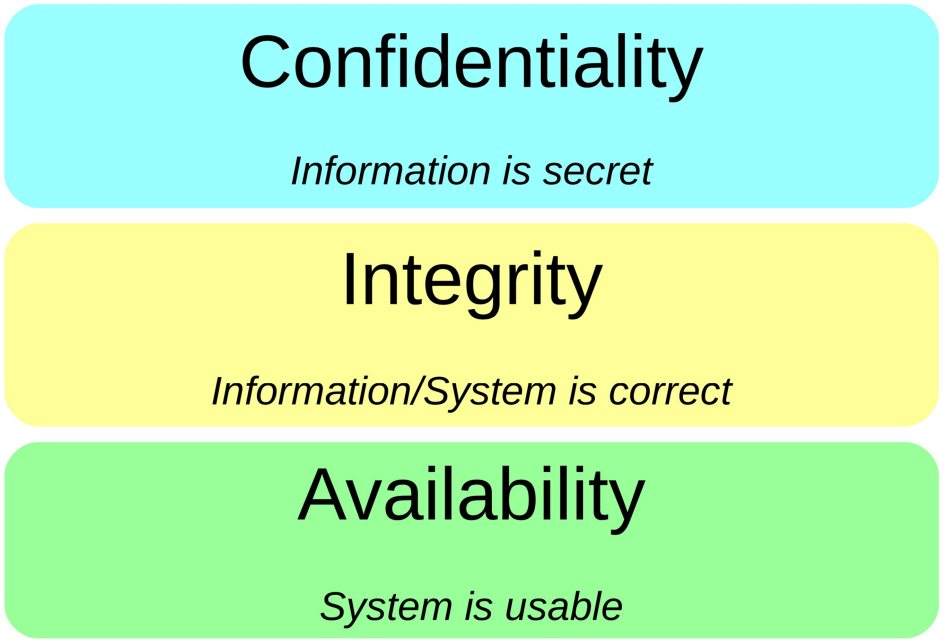
Hello there,

Martin here from AWS.

I'm happy to advise that 100% of the charges for the compromised activity on y
been waived. Rest assured, you no longer have to worry about the charges.

To avoid similar compromises in the future, please consider the following to he
security of your account.

What's Security?



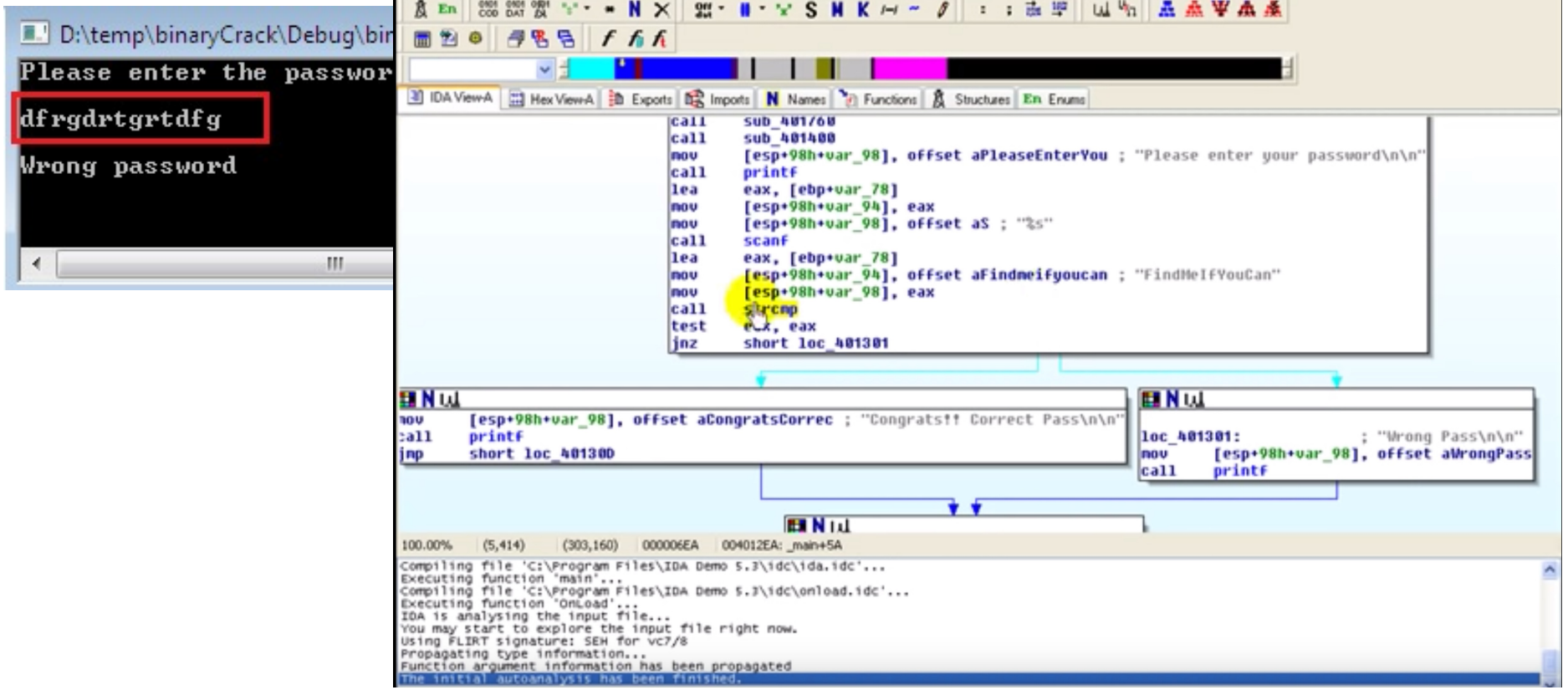
You will again and again come to this part when doing security research…

Typical Topics Covered in Cybersecurity Studies

- Security basics and principles :
  - Confidentiality, integrity, availability, attack models
- Cryptography:
  - Basic crypto primitives, public key crypto, signatures, authentication, symmetric crypto
- Software security:
  - Memory errors, buffer overflow, obfuscation, malware, security testing
- System & web security:
  - Authentication, access control, protocols, browser security, side channel attacks
- Security on emerging platforms:
  - blockchain; IoT; AI;

Reverse Engineering

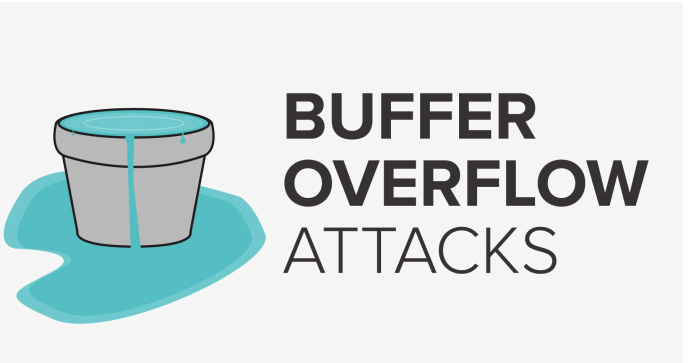
- How to break the password protection of a Windows software?



How to find vulnerabilities in software?

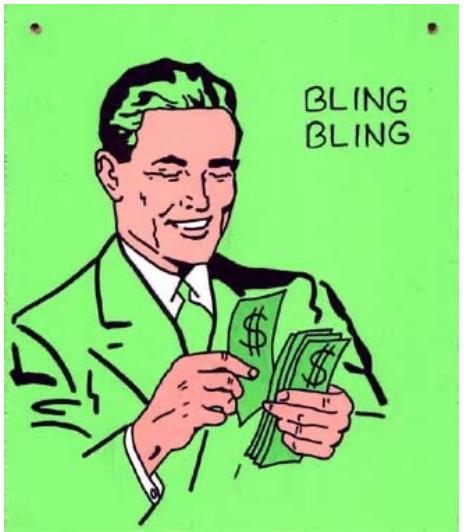
```
#include <stdio.h>
#define MAX_IP_LENGTH 15
int main(void) {
  char file_name[] = "ip.txt";
  FILE *fp;
  fp = fopen(file_name, "r");
  char ch;
  int counter = 0;
  char buf[MAX_IP_LENGTH];
  while((ch = fgetc(fp)) != EOF) {
    buf[counter++] = ch;
  }
```

Buffer overflow if “ip.txt” has more than 15 bytes.



Vulnerability Finding Today

- Security bugs can bring $500-$100,000 on the open market
- Good bug finders make $180-$250/hr consulting
- Few companies can find good people, many don't even realize this is possible.
  - Google: Team Zero; Tencent: Keen Lab; …
- Still largely a black art



Automatic Vulnerability Detection

Find a needle in a haystack

OSS-Fuzz - continuous fuzzing for open source software.

google.github.io/oss-fuzz

Fuzz Testing

CodeQL

Discover vulnerabilities across a codebase with CodeQL, our industry-leading semantic code analysis engine. CodeQL lets you query code as though it were data. Write a query to find all variants of a vulnerability, eradicating it forever. Then share your query to help others do the same.

Information flow analysis

Formal Verification

- Formal verification can (ideally) completely eliminate vulnerabilities.
  - Mathematically prove the absence of bugs.
- How to I know the insertion sort will always return a sorted list?

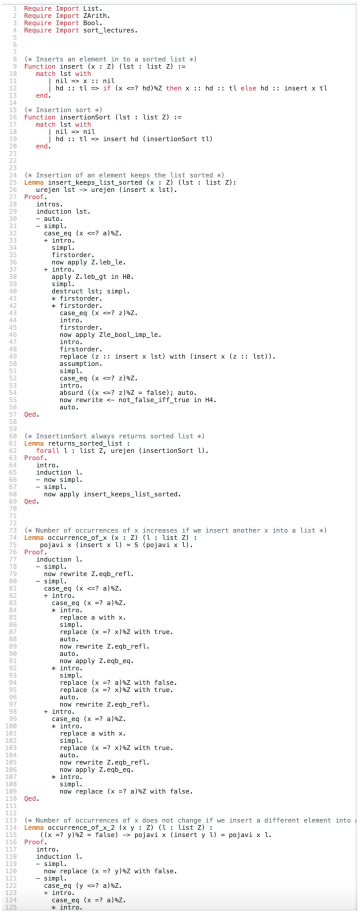
```
# Python program for implementation of Insertion Sort

# Function to do insertion sort
def insertionSort(arr):

    # Traverse through 1 to len(arr)
    for i in range(1, len(arr)):

        key = arr[i]

        # Move elements of arr[0..i-1], that are
        # greater than key, to one position ahead
        # of their current position
        j = i-1
        while j >=0 and key < arr[j] :
                arr[j+1] = arr[j]
                j -= 1
        arr[j+1] = key


# Driver code to test above
arr = [12, 11, 13, 5, 6]
insertionSort(arr)
print ("Sorted array is:")
for i in range(len(arr)):
    print ("%d" %arr[i])
```

Formal Verification

- You can prove it, as how you prove some Euclidean geometry properties.

```python
# Python program for implementation of Insertion Sort

# Function to do insertion sort
def insertionSort(arr):

    # Traverse through 1 to len(arr)
    for i in range(1, len(arr)):

        key = arr[i]

        # Move elements of arr[0..i-1], that are
        # greater than key, to one position ahead
        # of their current position
        j = i-1
        while j >=0 and key < arr[j] :
                arr[j+1] = arr[j]
                j -= 1
        arr[j+1] = key


# Driver code to test above
arr = [12, 11, 13, 5, 6]
insertionSort(arr)
print ("Sorted array is:")
for i in range(len(arr)):
    print ("%d" %arr[i])
```

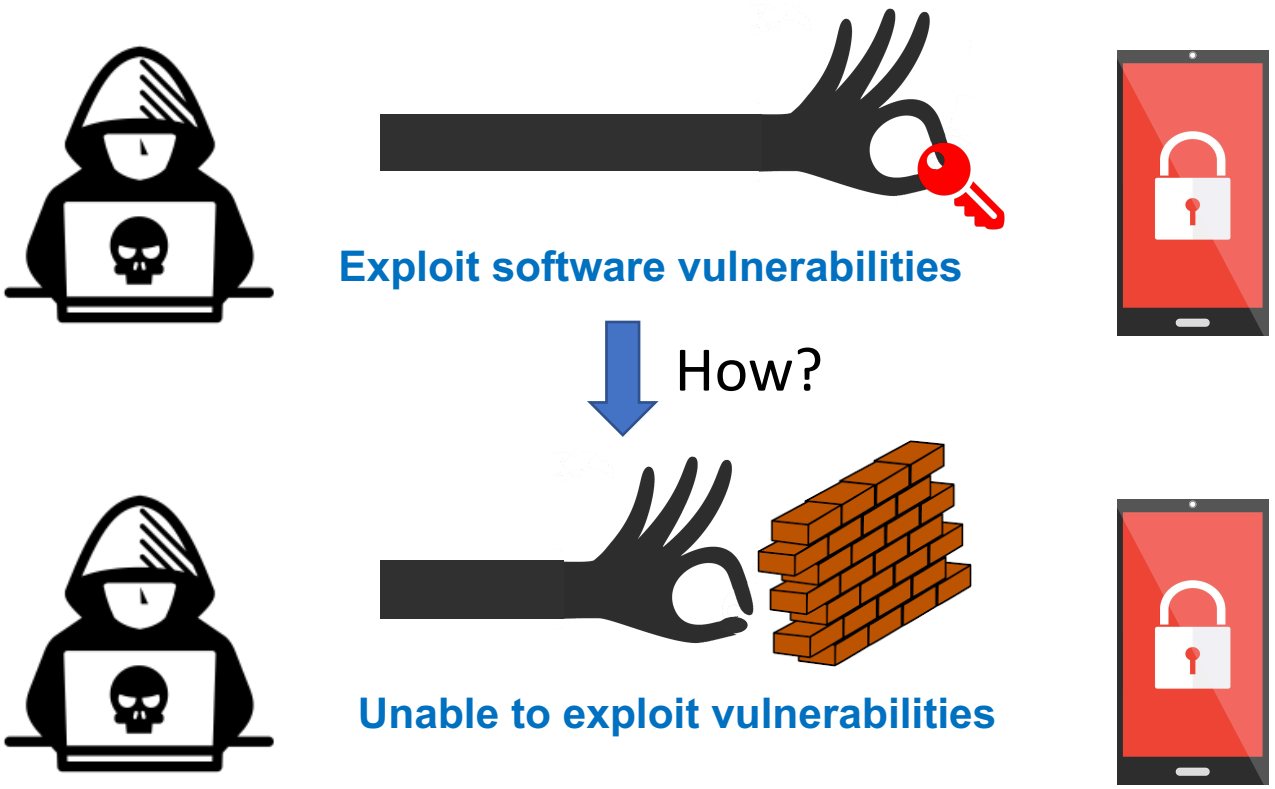
Proof



Computer will check the correctness

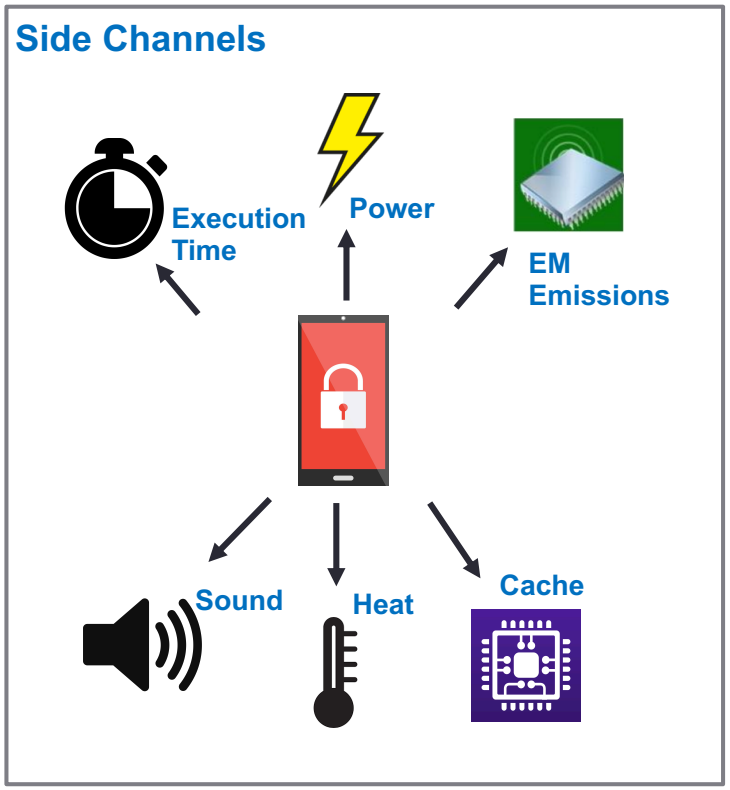
Haven’t finished yet..

Side Channel Attacks



Side Channel Attacks

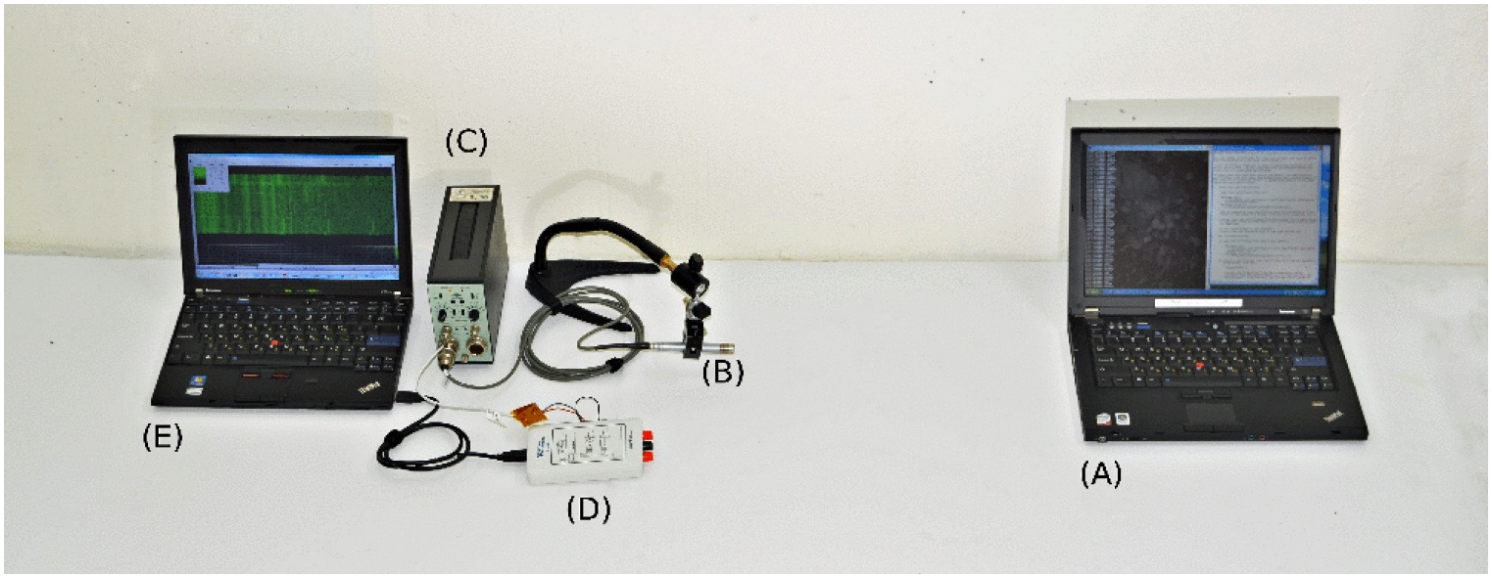
- De-facto exploitations in Cybersecurity



Infer secrets via **secret-dependent** physical information.

Side Channel Attacks

- Infer your secrets (password; private key) via acoustic side channel attack

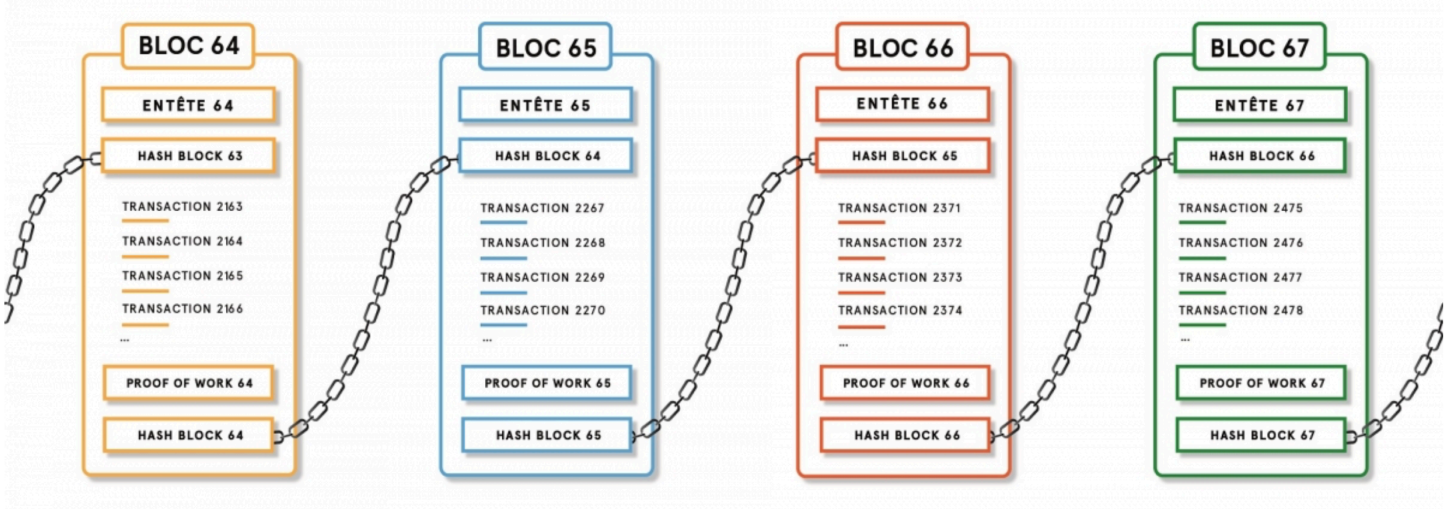


Attacker's

Victim's

Blockchain

The best real-world crypto application ~~and have made many millionaires~~?



Bitcoin

- Unregulated digital currency
- Bitcoin transactions are stored on Blockchain
- Each anonymous address on the blockchain acted as a simple bank account.

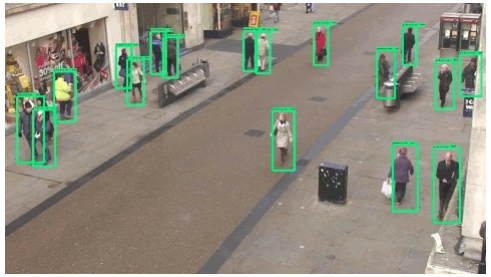
Ethereum

- Unregulated digital currency and computing system
- Smart contracts: programs executed on the blockchain
- Each anonymous address on the blockchain could be a user or a smart contract.

Blockchain figure from: https://blog.theodo.com/2018/01/deploy-first-ethereum-smart-contract-blockchain/

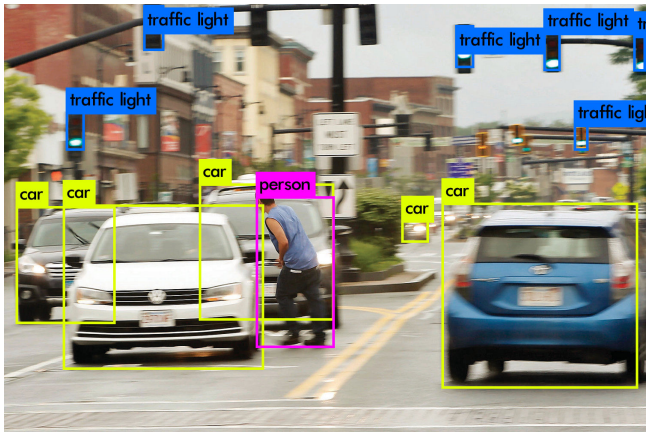
Artificial Intelligence

- AI techniques have been used for security purposes.

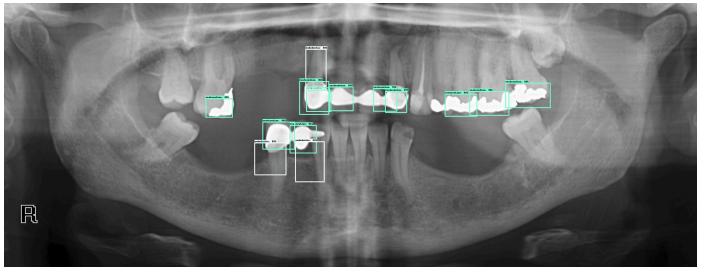


Surveillance Camera

Surveillance Camera



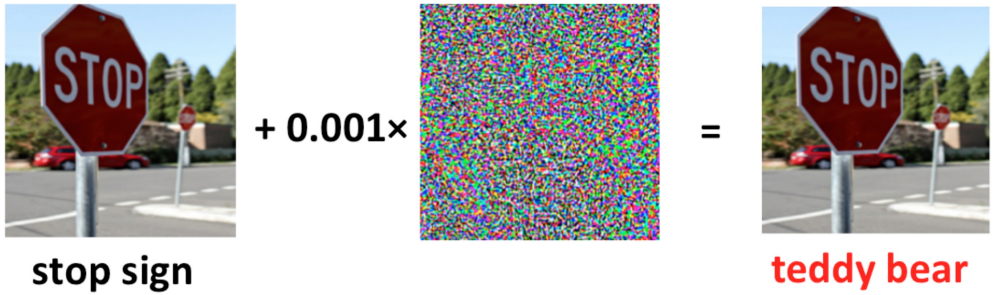
Auto-Driving Systems



Medical Image Processing

Artificial Intelligence

- Adversarial attacks are popular...



Classification failure

Object detection failure

We will talk more cases on AI security.

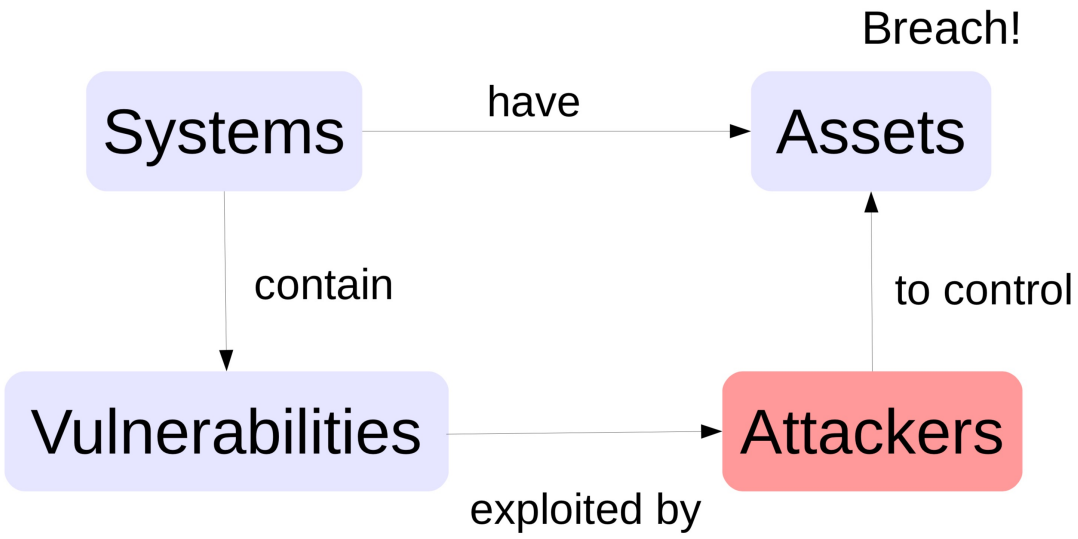
The Security Mindset

Attacker vs. defender

The Security Mindset

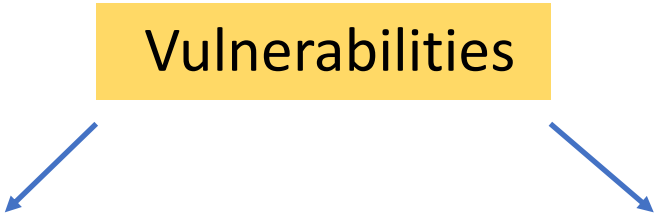
- Think like a cyber attacker
  - Understand techniques and opportunities for exploiting security. → next two slides
- Think like a cyber defender
  - Know yourself: security policy
  - Know yourself: risk assessment
  - Know your enemy: threat model
  - Benefits vs. costs:
    - Some security defenses are just too expensive

Think Like an Attacker



Think Like an Attacker

Where do vulnerabilities come from?



Design

- Theoretical limitations
- Lack of security features (e.g. authentication)
- Side channels
- Wrong threat model
- Wrong user model

Implementation

- Coding errors
- Hardware errors
- Injected errors
- Failure to meet specifications

Where can I get vulnerabilities?

- Find unknown vulnerabilities → hard
- Buy zero-day vulnerabilities → well, you can do that..?
- Reuse known vulnerabilities → easy?

But Why Good Citizens Need to Know How to Attack?

To understand this, think about why biologists would study (unknown) virus...

White hat wizards!

- Identify vulnerabilities so they can be fixed.
- Learn about unknown threats.
- Help venders to build more secure systems.
- ~~And get lots of bonus from vendors~~

Think Like a Defender

- Security policy
  - What property we are trying to enforce?
    - E.g., password can only be stored within my phone.
    - E.g., data pointers in your C code can only access certain memory region.
  - Could be difficult to even define the policy/specification
- Risk assessment
  - Identify assets (e.g., network, servers, applications, data centers, etc.) within the organization.
  - Asset criticality.
  - Measure the risk ranking for assets and prioritize them for assessment.

Think Like a Defender

- Threat model
  - Who are the attackers?
  - What kind of capability they have?
  - What kind of information/data they try to steal?

Think Like a Defender

- Threat model for a (simplified) cloud computing platform
  - Attacker; capability; assets

Think Like a Defender

- Threat model
  - Who are the attackers?
    - Service provider, and other users
  - What kind of capability they have?
    - Service provider can control anything
    - Attackers on the cloud VM can share the same hardware with you
      - Common threat model for side channels
  - What kind of assets they try to steal?
    - Anything valuable!

Think Like a Defender

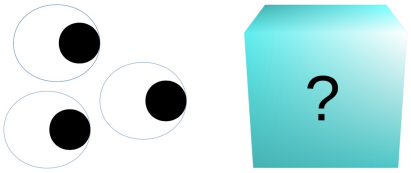
- Costs vs. benefits?
  - For example, to protect an OS kernel from being exploited, you can have two options:
    - Online monitoring:
      - easy to do.
      - slow down the performance
    - Offline formal verification:
      - very difficult to conduct for commercial OS.
      - But no penalty for online performance.
- Saltzer and Schroeder's Principles of Secure Design
  - A series of design principles for secure systems
    - Extensions for reading after the class.
  - Some of the rules may not be applicable nowadays.

Saltzer and Schroeder's Principles of Secure Design

- 1) Open Design vs. Obscure Design

The system's design should be openly available to everyone.

"Given enough eyeballs, all bugs are shallow"
-- Linus Torvalds



"The eyeballs weren't looking"

Saltzer and Schroeder's Principles of Secure Design

- 2) Economy of Mechanism

The system should be simple enough to understand and analyze.

Helpful for security analysis:
- Debugging/code audit
- Static/dynamic analysis
- Formal verification

Clean interfaces between modules, avoid global state, etc.

Saltzer and Schroeder's Principles of Secure Design

- 3) Least Privilege

A subject should only be given the minimum necessary privileges for completing its task.

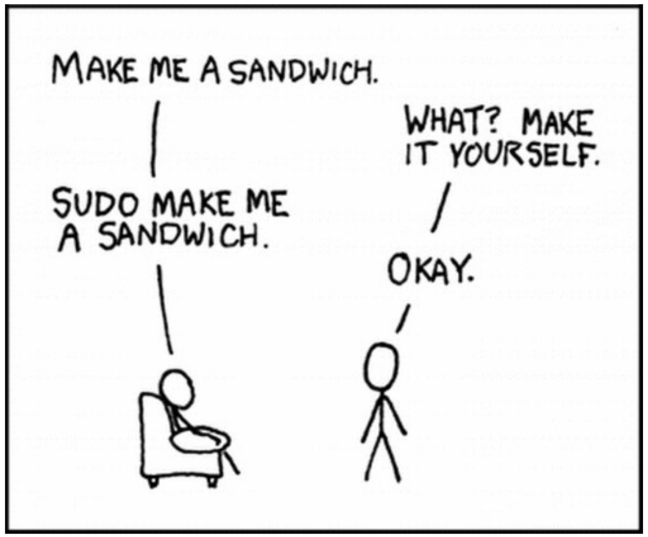


Figure out exactly what capabilities a program requires in order to run, and grant exactly those

- This is not easy. One approach is to start with granting none, and see where errors occur.

Summary

- The endless arms race between cyber attackers and defenders lead to many interesting problems
  - For doing research & engineering
- Be a happy and ethic hacker!
  - Otherwise, you (and your teacher) might run into trouble …