Lec 2: The Basics

CS492E: Introduction to Software Security

Sang Kil Cha

What is Computer Security?

Is Your Computer Secure?

In which condition can you say your computer is secure?

Secure Systems Satisfy the CIA Properties

- **C**onfidentiality: information is not made available to unauthorized parties.
- **I**ntegrity: information is not modified in an unauthorized manner.
- **A**vailability: information is readily available when it is needed.

Implication of Availability

Imagine a unbreakable and unopenable vault. Is it useful?

Security vs. Privacy

Security is the ***state*** of being safe, but privacy is the ***control*** that you have over your personal information.

Security vs. Privacy

Security is the ***state*** of being safe, but privacy is the ***control*** that you have over your personal information.

- Company X removed Alice's name from the receipt to protect her ___.
- Company Y adopted a 2FA system to enhance the ___ of its web server.

Example: Perfect Secrecy?

Let p be a plaintext message. We create a random string of the same size and denote it as a key k. We then simply obtain a ciphertex message c by xoring each character of p and k: $c = p \oplus k$. Note that k will be used only once and never be reused.

Example: Perfect Secrecy?

Let p be a plaintext message. We create a random string of the same size and denote it as a key k. We then simply obtain a ciphertex message c by xoring each character of p and k: $c = p \oplus k$. Note that k will be used only once and never be reused.

Is this secure?

Example: Secure Cipher

F# Implementation

```
let p = "software security"
let k = "ABCDEF01234567890"
let (^) (a: string) (b: string) = // element-wise XOR
  (a, b)
  ||> Seq.map2 (fun a b -> char (int a ^^^ int b))
  |> Seq.toArray
  |> System.String
let c = p ^ k // cipher text
let p' = c ^ k // plain text
"random value" ^ k // ?
```

Advanced Topic

Can you imagine cases where only one or two of the CIA properties are satisfied, while being still “secure”?

Confidentiality vs. Secrecy

Often used interchangeably.

When you say something is ***secret***, it is kept hidden by you and it is yours.

When you say something is ***confidential***, it is not yours and it is only accessible by authorized entities (e.g., banks keep account information confidential).

Integrity vs. Authenticity

Integrity is about ensuring the data is ***unmodified***, and authenticity is about ensuring the data is ***originated*** from the claimed sender.

Integrity vs. Authenticity

Integrity is about ensuring the data is ***unmodified***, and authenticity is about ensuring the data is ***originated*** from the claimed sender.

> It is often the case that authenticity implies integrity.

How to Achieve Computer Security?

Cryptography

Cryptography is about ***confidentiality*** and ***integrity***.

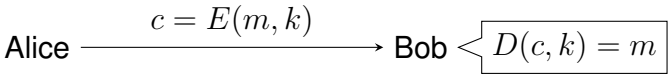
What about availability?

Cryptographic Primitives

- Symmetric key encryption/decryption.
- Public key encryption/decryption.
- Digital signatures.
- Hash functions.
- etc.

Symmetric Key Ciphers

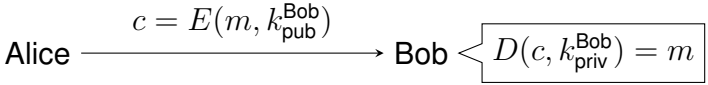
The same key k is used to encrypt/decrypt messages.



- Pros
  - Fast.
  - Intuitive.
- Cons
  - Key sharing is difficult.
  - Digital sign is difficult.
  - Once the key is compromised, then the whole system becomes useless.

Asymmetric Key Ciphers

Each party has two distinct keys: public k_{pub} and private key k_{priv}.



- Pros?
- Cons?

Digital Signature

The goal = ***authenticity*** + non-repudiation.

Non-repudiation: once a message is signed, then the principal cannot deny that the message is signed by the principal.

Digital signing is trivial with asymmetric crypto: encrypt with a private key to sign a message, and decrypt with the public key to verify the signature.

Cryptographic Hash Function

Hash functions that are “difficult” to break: preimage resistance, collision resistance, etc. Typically, hashing is orders of magnitude faster than encryption.

Verifying the integrity of a message with hash?

- Send m and $m' = h(m)$.
- Receiver verifies the integrity of m by computing $h(m) = m'$.

Our Goal

Cryptographic primitives we learned:

- Symmetric key encryption/decryption.
- Public key encryption/decryption.
- Digital signatures.
- Hash functions.

Use/combine those cryptographic primitives to build a secure system.

Example: Password Storage

- Goal: store ID and password pairs to authenticate users.
- Bad approach: store ID and password pairs in plaintext to a DB.

ID	Password
alice	1234abcd
bob	verysecure
charlie	1234abcd

Example: Password Storage (Improvement)

- (Option 1) Encrypt every password before storing it to the DB.
- (Option 2) Hash every password before storing it to the DB.

Example: Password Storage (Improvement)

- (Option 1) Encrypt every password before storing it to the DB.
- (Option 2) Hash every password before storing it to the DB.

ID	Hash
alice	$h(\text{1234abcd})$
bob	$h(\text{verysecure})$
charlie	$h(\text{1234abcd})$

Is Hash-based Password Storage Secure?

Assume that the DB is stolen by an attacker. What can the attacker do?

Is Hash-based Password Storage Secure?

Assume that the DB is stolen by an attacker. What can the attacker do?

Just brute-forcing, but what about common passwords?

Salted Hash

Use a randomly generated number (a salt) to make a hash.

ID	Salt	Hash
alice	42	$h(\text{1234abcd}, 42)$
bob	3	$h(\text{verysecure}, 3)$
charlie	99	$h(\text{1234abcd}, 99)$

Many Other Secure Applications

- SSH: Secure shell.
- HTTPS/TLS: Web.
- WPA2/WPA3: Wifi.
- Bitcoin, blockchain, etc.

How about Software Security?

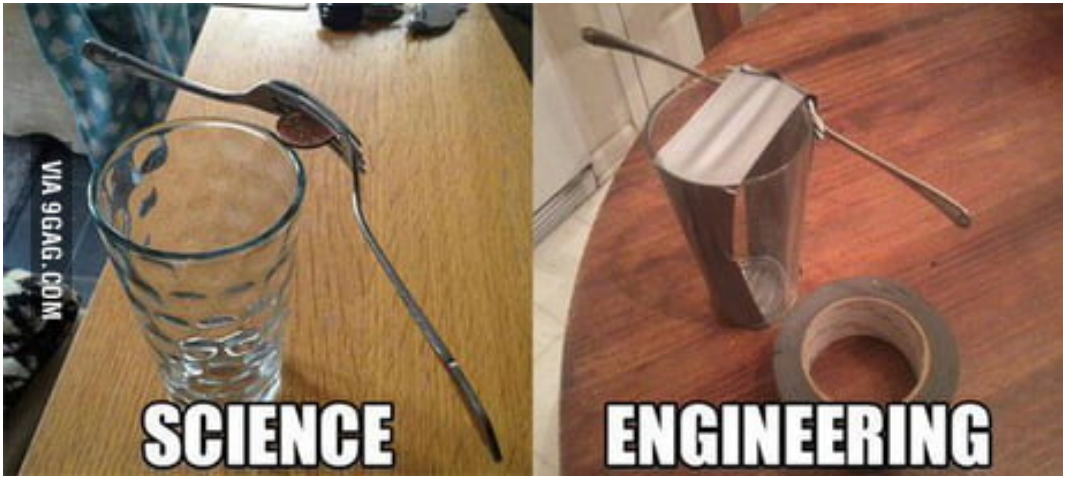
Secure Software?

Can we say a program is secure if it satisfies the CIA properties?

What is Missing?

- Theory vs. Practice.
- Science vs. Engineering.

Science vs. Engineering?



Software is Buggy

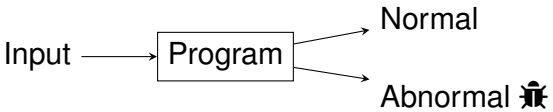
Bugs are plentiful.

- About 15 - 50 bugs per 1000 lines of code[1].
- Regardless of the languages used.

[1] Code Complete, Microsoft Press, 2004

Software Bug?

Software bug is an error/fault/mistake in the code that produces an unexpected result.



Software Bug is the Key

Root cause of many security problems including malware, hacking incidents, phishing, privacy leakage, etc.

Not Every Bug is Security Critical

Some bugs are merely about aesthetics. Some bugs allow an attacker to compromise the whole system.

Therefore, you should develop an eye for figuring out which bugs are critical or not. You should ***think like adversaries***!

Think Like Adversaries

Threat Modeling

Threat modeling is the process of systematically identifying threats to a system, such as vulnerabilities or lack of defense mechanisms.

Attack Scenario

This course is too difficult to follow. However, you really should get an ‘A’ in this course in order to graduate.

Of course you should never attack anyone in reality!

Potential Threats

1. Steal the solutions.
2. Hire security experts to solve exams on behalf of you.
3. Modify my score after grading is done (get administrative privilege from the KAIST academic system).
4. Perform denial of service attack on KLMS.
5. etc.

Depending on how much effort you are willing to put ...

Enumerate Attack Points

To “steal the solution” what do you have to do? Where would be the solutions?

- Prof. Cha’s laptop.
- Prof. Cha’s desktop.
- Prof. Cha’s GitHub account.
- The trash bin in Prof. Cha’s office.
- TA’s emails.
- TA’s office.
- KLMS.
- etc.

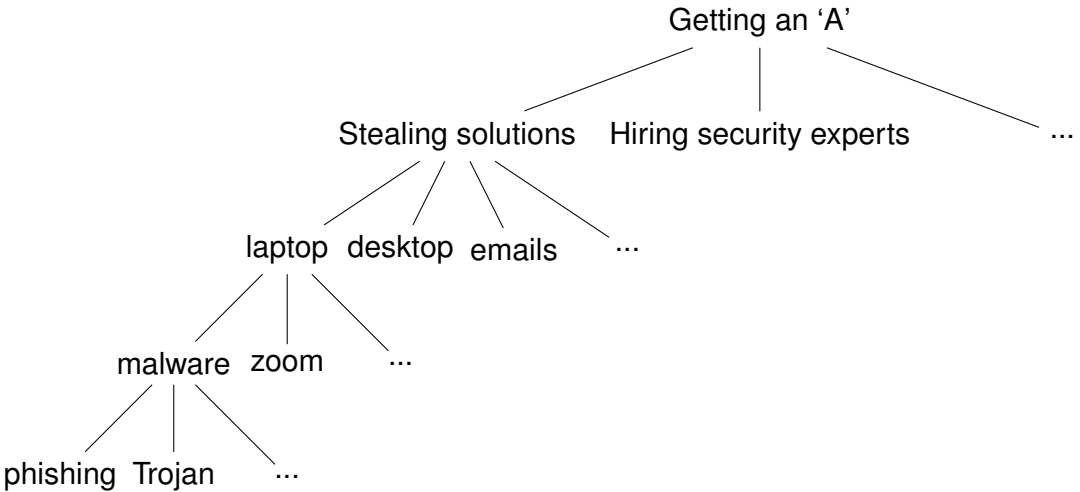
Enumerate Attack Points

To “steal the solution” what do you have to do? Where would be the solutions?

- Prof. Cha’s laptop.
- Prof. Cha’s desktop.
- Prof. Cha’s GitHub account.
- The trash bin in Prof. Cha’s office.
- TA’s emails.
- TA’s office.
- KLMS.
- etc.

Oftentimes, those attack points are called the ***Attack Surface***

Attack Tree


Getting an ‘A’
Stealing solutions
Hiring security experts
...
laptop
desktop
emails
...
malware
zoom
...
phishing
Trojan
...

Conclusion

What should you do now in order to make your software/information/computer secure?

- Learn how to use basic cryptographic primitives (next lecture).
- Learn how to spot/fix critical bugs in software.

Question?