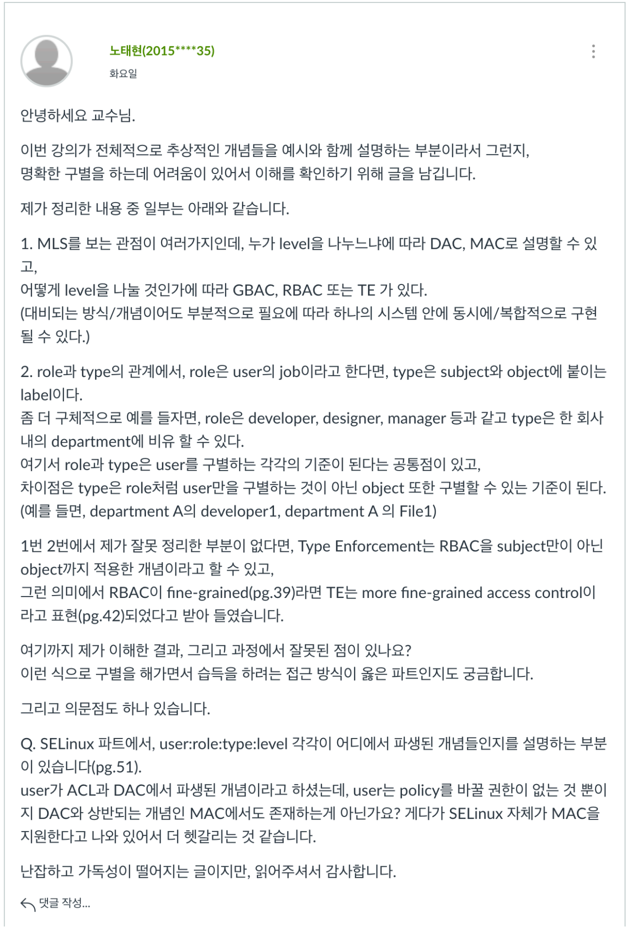
SWE3025: Computer Security
Lecture 0x05: Crypto and TLS

Hojoon Lee

Systems Security Lab @ SKKU

Answering Your Voice

Answering Your Voice



노태현(2015****35)
화요일

안녕하세요 교수님.

이번 강의가 전체적으로 추상적인 개념들을 예시와 함께 설명하는 부분이라서 그런지,
명확한 구별을 하는데 어려움이 있어서 이해를 확인하기 위해 글을 남깁니다.

제가 정리한 내용 중 일부는 아래와 같습니다.

1. MLS를 보는 관점이 여러가지인데, 누가 level을 나누느냐에 따라 DAC, MAC로 설명할 수 있고,
어떻게 level을 나눌 것인가에 따라 GBAC, RBAC 또는 TE 가 있다.
(대비되는 방식/개념이어도 부분적으로 필요에 따라 하나의 시스템 안에 동시에/복합적으로 구현될 수 있다.)

2. role과 type의 관계에서, role은 user의 job이라고 한다면, type은 subject와 object에 붙이는 label이다.
좀 더 구체적으로 예를 들자면, role은 developer, designer, manager 등과 같고 type은 한 회사 내의 department에 비유 할 수 있다.
여기서 role과 type은 user를 구별하는 각각의 기준이 된다는 공통점이 있고,
차이점은 type은 role처럼 user만을 구별하는 것이 아닌 object 또한 구별할 수 있는 기준이 된다.
(예를 들면, department A의 developer1, department A 의 File1)

1번 2번에서 제가 잘못 정리한 부분이 없다면, Type Enforcement는 RBAC을 subject만이 아닌 object까지 적용한 개념이라고 할 수 있고,
그런 의미에서 RBAC이 fine-grained(pg.39)라면 TE는 more fine-grained access control이라고 표현(pg.42)되었다고 받아 들였습니다.

여기까지 제가 이해한 결과, 그리고 과정에서 잘못된 점이 있나요?
이런 식으로 구별을 해가면서 습득을 하려는 접근 방식이 옳은 파트인지도 궁금합니다.

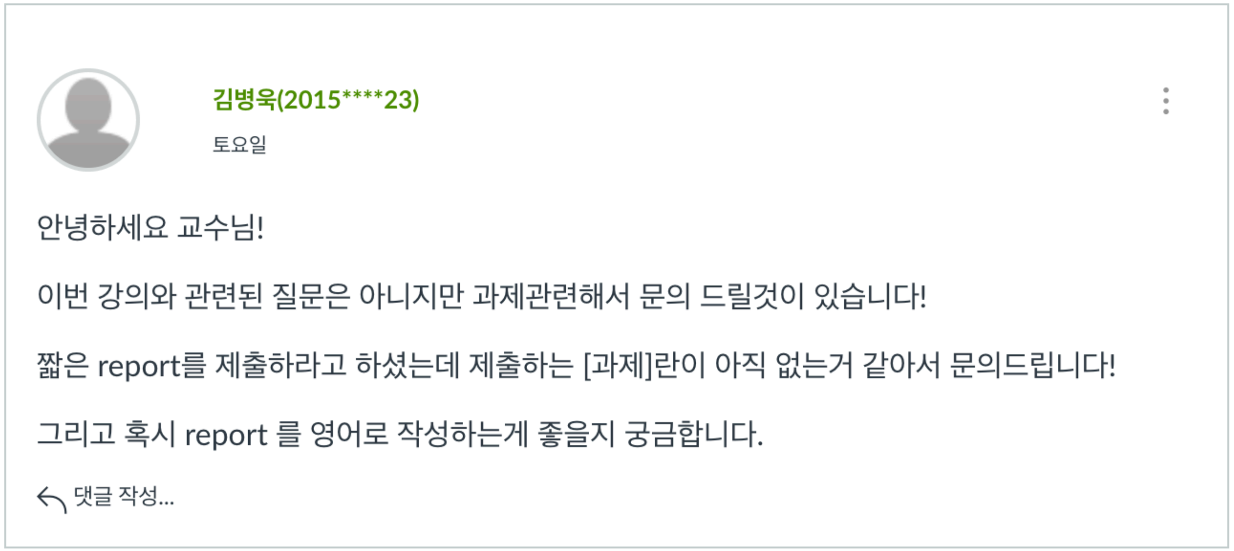
그리고 의문점도 하나 있습니다.

Q. SELinux 파트에서, user:role:type:level 각각이 어디에서 파생된 개념들인지를 설명하는 부분이 있습니다(pg.51).
user가 ACL과 DAC에서 파생된 개념이라고 하셨는데, user는 policy를 바꿀 권한이 없는 것 뿐이지 DAC와 상반되는 개념인 MAC에서도 존재하는게 아닌가요? 게다가 SELinux 자체가 MAC을 지원한다고 나와 있어서 더 헷갈리는 것 같습니다.

난잡하고 가독성이 떨어지는 글이지만, 읽어주셔서 감사합니다.

댓글 작성...

Answering Your Voice



김병욱(2015****23)

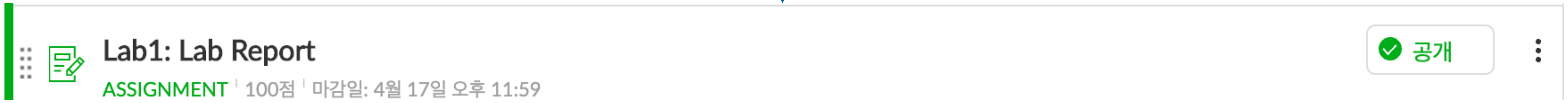
토요일

안녕하세요 교수님!

이번 강의와 관련된 질문은 아니지만 과제관련해서 문의 드릴것이 있습니다!

짧은 report를 제출하라고 하셨는데 제출하는 [과제]란이 아직 없는거 같아서 문의드립니다!

그리고 혹시 report 를 영어로 작성하는게 좋을지 궁금합니다.



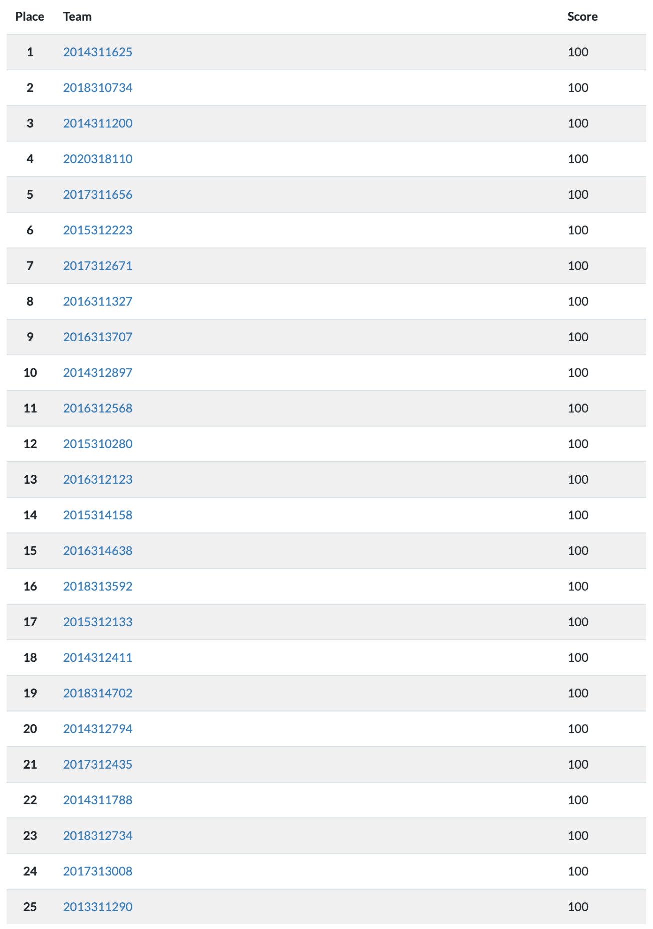
Lab1: Lab Report

ASSIGNMENT | 100점 | 마감일: 4월 17일 오후 11:59

CTF Challenge

- Lab1 CTF Challenge:

  Confused Deputy
  - Having fun?
  - Too easy? Too difficult?
  - Are you familiar with Linux/Unix environment?
  - How familiar are you with reverse engineering?



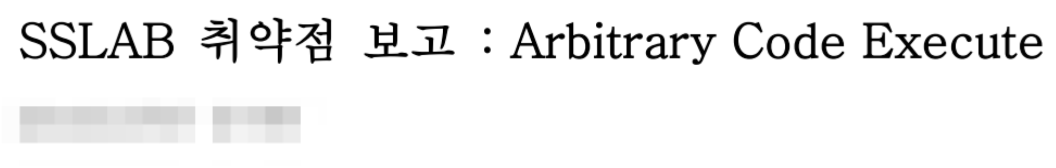
Place	Team	Score
1	2014311625	100
2	2018310734	100
3	2014311200	100
4	2020318110	100
5	2017311656	100
6	2015312223	100
7	2017312671	100
8	2016311327	100
9	2016313707	100
10	2014312897	100
11	2016312568	100
12	2015310280	100
13	2016312123	100
14	2015314158	100
15	2016314638	100
16	2018313592	100
17	2015312133	100
18	2014312411	100
19	2018314702	100
20	2014312794	100
21	2017312435	100
22	2014311788	100
23	2018312734	100
24	2017313008	100
25	2013311290	100

SSLab CTF Framework

- Written a significant portion of the SSLab CTF Framework
- "It's fun to work on projects like this" -Kha
- Round of applause for our TA

Kha Dinh Duy

SSLab CTF Framework


SSLAB 취약점 보고 : Arbitrary Code Execute

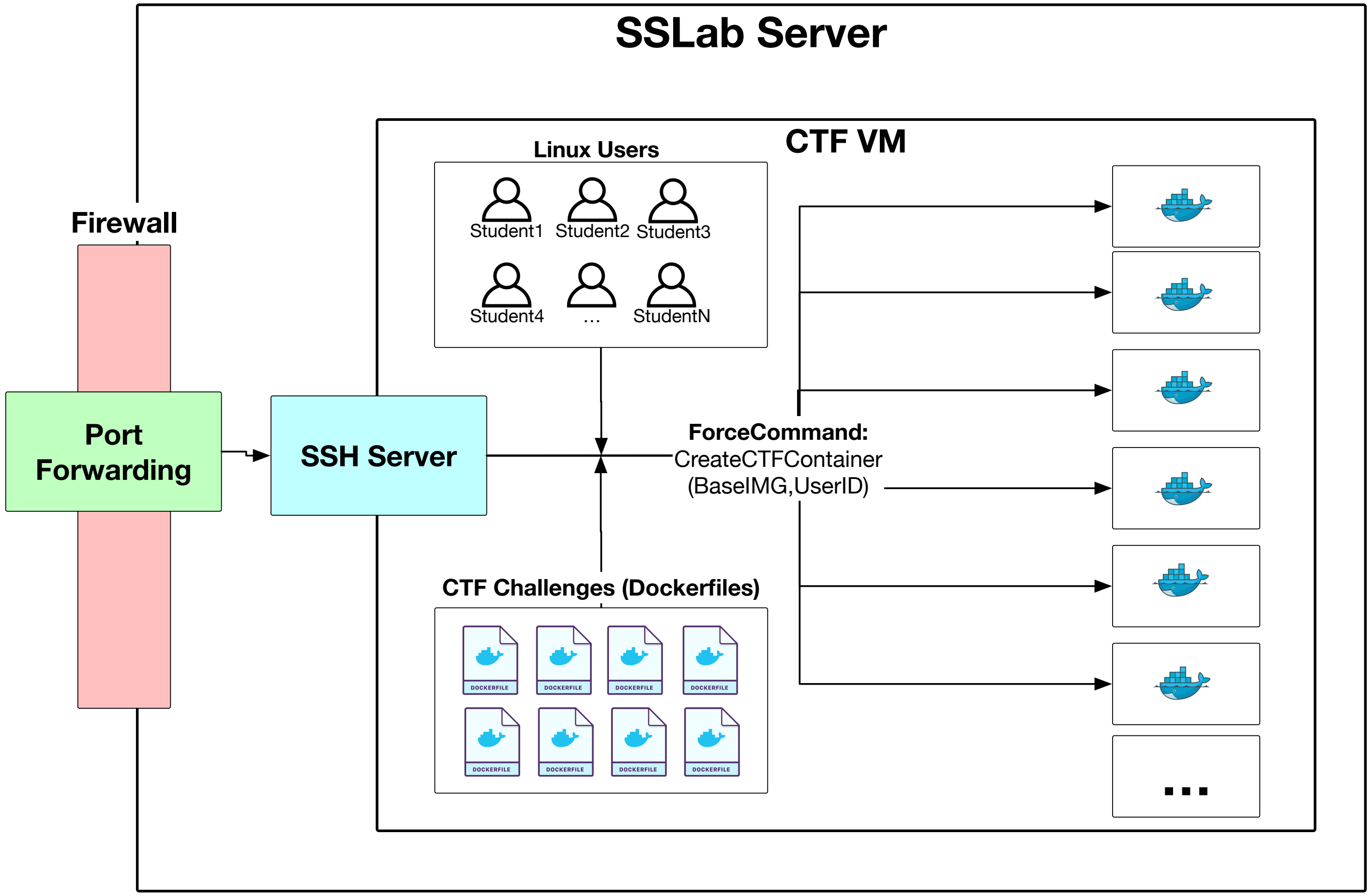
[redacted]

- A student whose name shall remain anonymous (I haven't got his/her consent yet)
- Reported a few possibly abusable security loopholes in our CTF framework
- We will not reveal the loopholes at this point to prevent abusing
- But the student gets +10 points in his lab1 assignment (but not above 100)

SSLab CTF Framework

- Open Design Principle
  - Hiding your security mechanism does not help you
  - Security mechanism should still be secure when its design is fully open
- Kerckhoff's principle of cryptography
  - A cryptosystem should be secure if everything about the system (except the key) is public knowledge

SSLab CTF Framework



SSLab CTF Framework

- Security Concern #1: How secure is SSH forcecommand?
  - You can force users to execute a certain command upon ssh connect
    - e.g., launch and enter docker container
  - The command is probably executed using a shell interpreter (e.g., /bin/bash)
  - Can you avoid entering the docker container and get shell somehow? a super fast CTRL-C as soon as ssh connects?
- Security Concern #2: Password == ID
  - This issue can be solved by using a Public Key (which we will learn in this lecture or next)

Crypto Concepts (From Security Engineer and Programmer Perspectives)

Objectives of this Lecture

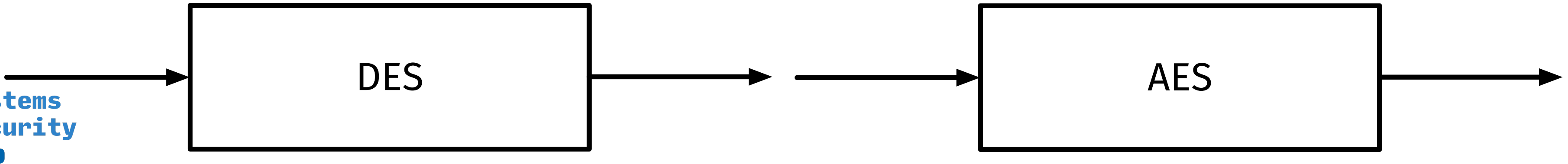
- Crypto Basics
- Symmetric Key Cryptography
- Asymmetric Key Cryptography (Public Key)
- Hash Functions, MAC, HMAC
- Shared secret Generation (Diffie-Hellman Key Exchange)
- Final Objective: How all of the above work together in SSL/TLS

Disclaimer

- In this course, you are not required to understand <u>cryptography-side</u> of the things we learn
- We will focus on understanding the <u>big picture</u>
  - How today's secure communication and protocols are built using crypto
  - How crypto algorithms are applied for integrity, confidentiality etc..

Disclaimer

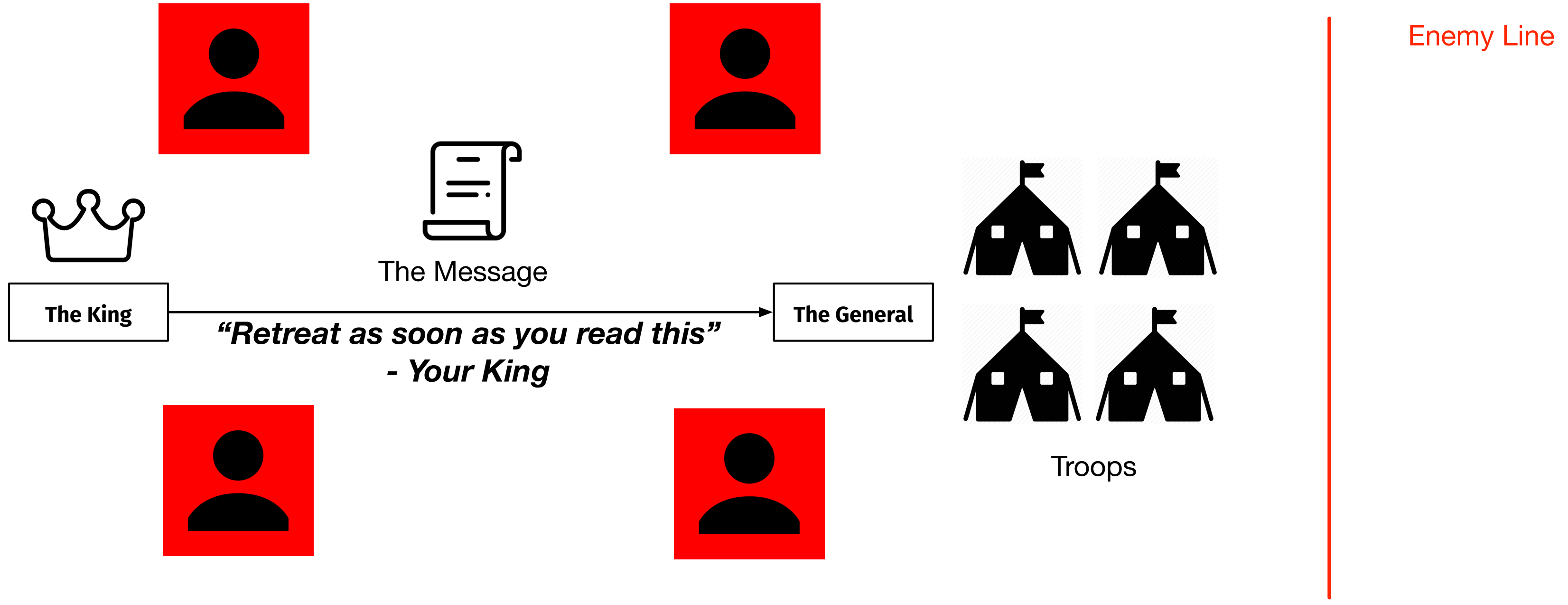
- Rationale: security engineers and programmers today use well-established crypto algorithms as <u>building blocks</u>
- In most cases, programmers must have a good understanding of <u>What they do</u> , but not necessarily <u>How</u> they work,



Crypto Overview

- The main objective of cryptography is to secure communication over insecure communication channels
- Security Goals
    - Confidentiality
    - Integrity
    - Authenticity

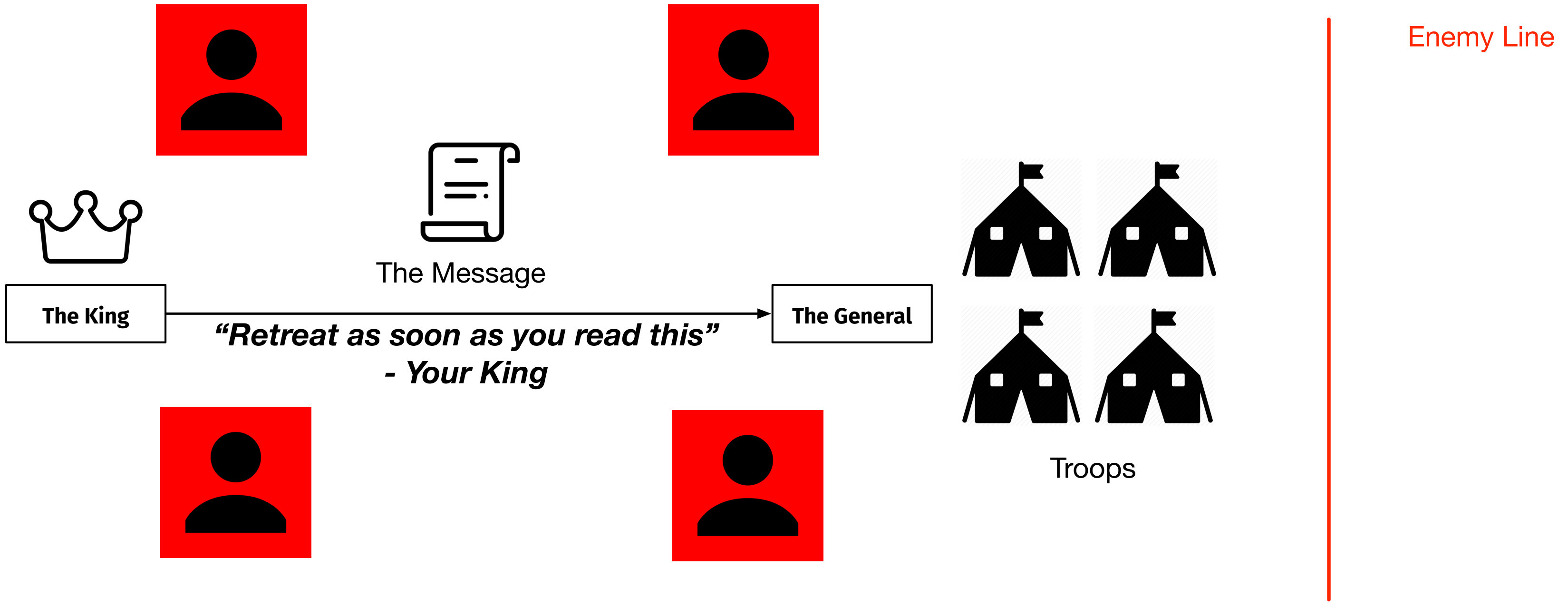
Overview



Confidentiality

- Even if our enemy captures the messenger and gets hold of the lettter,
- They must not be able to understand the message

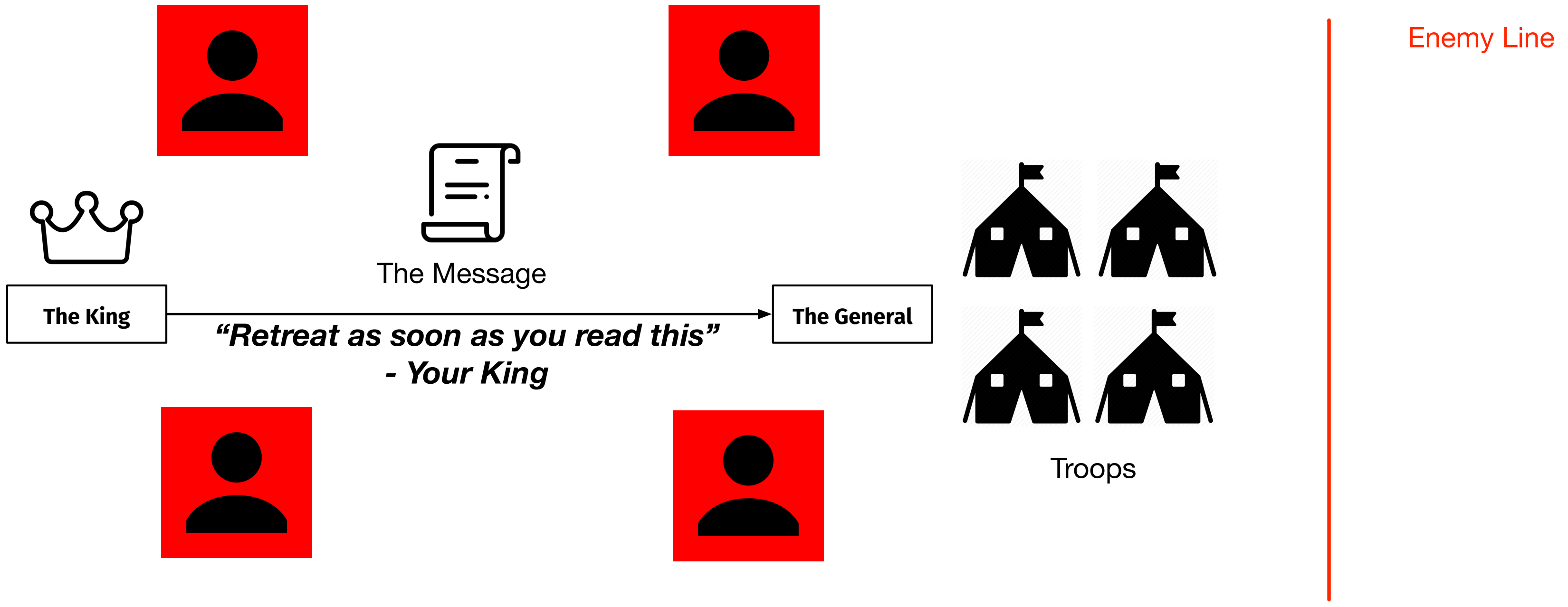
Overview



Integrity

- What if the message Had been stolen and already modified?
- What if the initial message was "You must hold the line until..."

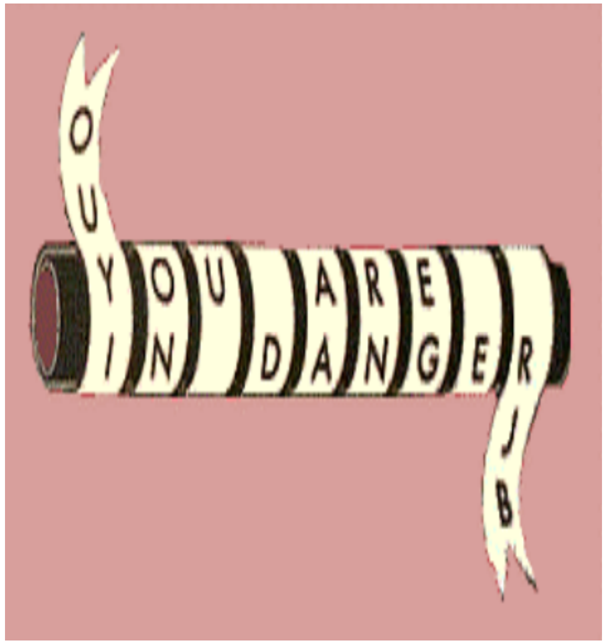
Overview



Authenticity

- How do we know if the message is even from our King?

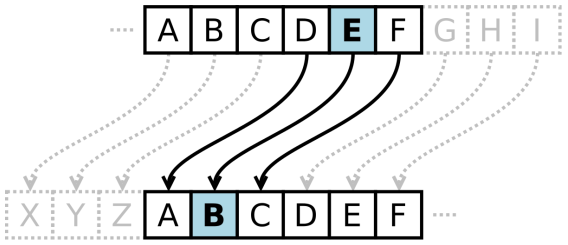
History and Cryptography



- In 405 BC, Greek general Lysander of Sparta received coded message
- The message was successfully decoded by wrapping it around a wooden baton

History and Cryptography

- 2000 years ago
  - Caesar Cipher
    - Shift each letter forward by a fixed number *n*
    - Encode and decode by hand
- During World War I,II
  - Mechanical devices for encrypting and decrypting messages
- Today
  - Modern cryptography: rely on mathematics and electronic systems



Crypto Terminology

- Cryptology — The art and science of making and breaking “secret codes”
- Cryptography — making “secret codes”
- Cryptanalysis — breaking “secret codes”
- Crypto — all of the above (and more)

How to Speak Crypto

- A *cipher* or *cryptosystem* is used to *encrypt* the *plaintext*
- The result of encryption is *ciphertext*
- We *decrypt* ciphertext to recover plaintext
- A *key* is used to configure a cryptosystem
- A *symmetric key* cryptosystem uses the same key to encrypt as to decrypt
- A *public key* cryptosystem uses a *public key* to encrypt and a *private key* to decrypt

Crypto

- Basic assumptions
  - The system is completely known to the attacker
  - Only the key is secret
  - That is, crypto algorithms are not secret
- This is known as Kerckhoffs' Principle
- (Crypto version of *Open Design Principle* in Saltzer and Schroeder)
- Why do we make such an assumption?
  - Experience has shown that secret algorithms tend to be weak when exposed
  - Secret algorithms never remain secret
  - Better to find weaknesses beforehand

Simple Substitution With Shifting

- A.K.A Caesar's cipher

Shift by k

a → c
b → d
c → e
…
z → b

Key k

$$c = E_k(\text{“abc”}),\ k = 2$$

$$= \text{“cde”}$$

Cryptanalysis of Caesar's Cipher

- We know that key is a fixed number *n*
- We know that the cipher algorithm E_k simply shifts each character by *n*
- How do we find the key?
  - Only 26 possible keys
  - Try them all – Exhaustive key search

Substitution Cipher With Permutation

Use any permutation of letters

a → e
b → c
c → q
…
z → a

Key permutation

$c = E_k(\text{“abc”})$

$= \text{“ecq”}$

Then $26! > 2^{88}$ possible keys!

Cryptanalysis of Permutation Cipher

- Possible number of keys are 26! > 2^{88} Keys
- Exhaustive Key Search???

Cryptanalysis of Permutation Cipher

- Possible number of keys are $26! > 2^{88}$ Keys
- Exhaustive Key Search???
- NO

Cryptanalysis of Permutation Cipher

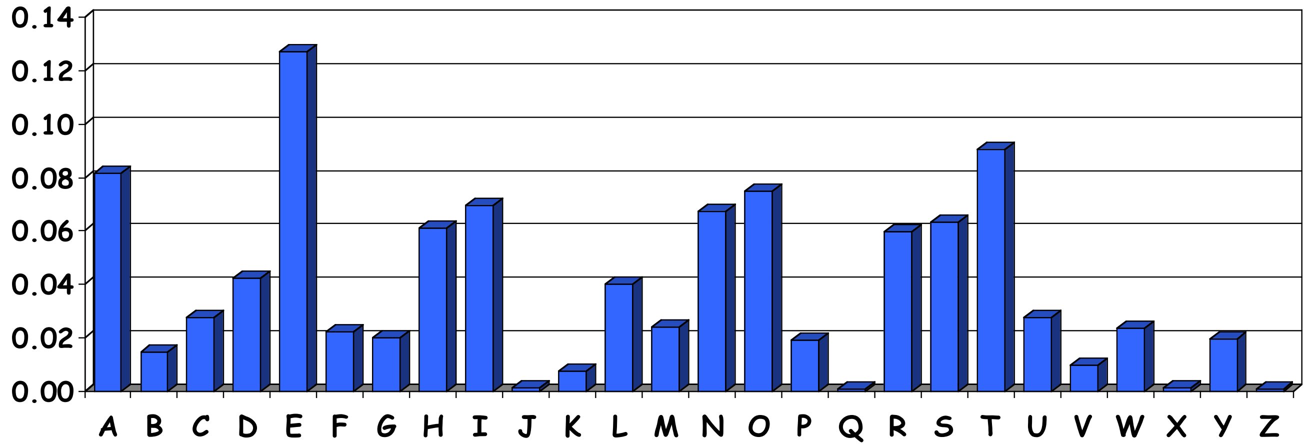
- Given Ciphertext:

  PBFPVYFBQXZTYFPBFEQJHDXXQVAPTPQJKTOYQWIPBVWLXTOXBTFXQWAXBVCXQWAXFQJVWLEQNT
  OZQGGQLFXQWAKVWLXQWAEBIPBFXFQVXGTVJVWLBTPQWAEBFPBFHCVLXBQUFEVWLXGDPEQVPQ
  GVPPBFTIXPFHXZHVFAGFOTHFEFBQUFTDHZBQPOTHXTYFTODXQHFTDPTOGHFQPBQWAQJJTODXQ
  HFOQPWTBDHHIXQVAPBFZQHCFWPFHPBFIPBQWKFABVYYDZBOTHPBQPQJTQOTOGHFQAPBFEQJ
  HDXXQVAVXEBQPEFZBVFOJIWFFACFCCFHQWAUVWFLQHGFXVAFXQHFUFHILTTAVWAFFAWTEVOITD
  HFHFQAITIXPFHXAFQHEFZQWGFLVWPTOFFA

- Is there any statistical/mathematical property that can be our "shortcut" to the plaintext?

Cryptanalysis II

- Cannot try all 2^{88} simple substitution keys
- Can we be more clever?
- English letter frequency counts...



Cryptanalysis II

- Ciphertext:

PBFPVYFBQXZTYFPBFEQJHDXXQVAPTPQJKTOYQWIPBVWLXTOXBTFXQWAXBVCXQWAXFQJVWLEQNTOZQGGQLFX
QWAKVWLXQWAEBIPBFXFQVXGTVJVWLBTPQWAEBFPBFHCVLXBQUFEVWLXGDPEQVPQGVPPBFTIXPFHXZHVFAG
FOTHFEFBQUFTDHZBQPOTHXTYFTODXQHFTDPTOGHFQPBQWAQJJTODXQHFOQPWTBDHHIXQVAPBFZQHCFWP
FHPBFIPBQWKFABVYYDZBOTHPBQPQJTQOTOGHFQAPBFEQJHDXXQVAVXEBQPEFZBVFOJIWFFACFCCFHQWAUV
WFLQHGFXVAFXQHFUFHILTTAVWAFFAWTEVOITDHFHFQAITIXPFHXAFQHEFZQWGFLVWPTOFFA

Ciphertext frequency counts:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
21	26	6	10	12	51	10	25	10	9	3	10	0	1	15	28	42	0	0	27	4	24	22	28	6	8

This is probably 'e' ???

Cryptanalysis: Terminology

- Cryptosystem is secure if best know attack is to try all keys
  - Exhaustive key search, that is
- Cryptosystem is insecure if *any* shortcut attack is known
- But then insecure cipher might be harder to break than a secure cipher!
  - What the … ?