Security

his

목차

- 암호화
- TLS
- 공격/방어

목표

- Confidentiality : 보내는 사람과 받는 사람만 내용을 알 수 있어야 한다.
- Integrity : 내용이 바뀌지 않아야 한다.
- Availability : 서비스는 항상 접근이 가능해야 한다
- Authenticate : 서로의 신분을 확인할 수 있어야 한다.

암호화

- 데이터를 원하는 사람만 이해할 수 있도록 변형시키는 것

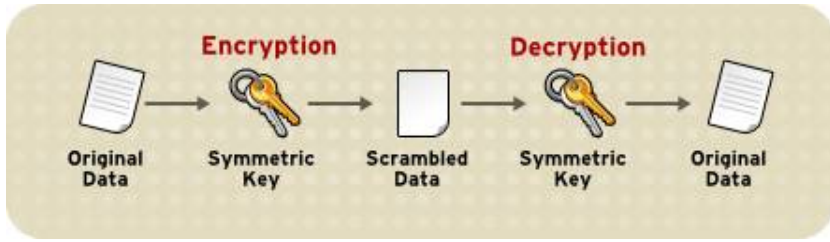
- 키 생성 -〉 암호화 -〉 복호화

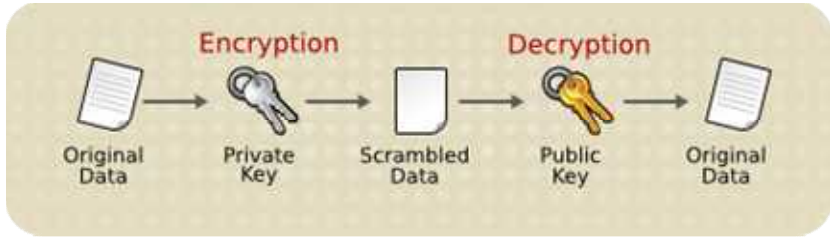
암호화 – 용어

- Plaintext (평문) : 암호화되지 않은 데이터
- Ciphertext (암호문) : 암호화한 데이터
- Key : 암호화할 때 필요한 정보
- Alice, Bob(송/수신자,A/B) Trudy(intruder) …

암호화 - 종류

- 대칭 키
  - 암호화와 복호화에 같은 키를 이용
- 비대칭 키
  - 암호화와 복호화에 다른 키를 이용
- 해싱
  - 암호화는 아님 - 완전히 변형





암호화 – 대칭 키 vs 비대칭 키

- 속도 : 대칭 키 〉 비대칭 키
- 키 분배 : 대칭 키 〈 비대칭 키
  - 대칭 키는 N명이 각각 모두 연결하는 상황 – N(N-1)/2
  - 비대칭 키는 각각은 private key만 갖고, public은 그냥 공개 – N
  - https://www.facebook.com/media/set/?set=oa.993976753976194&type=1

대칭 키 - stream cipher

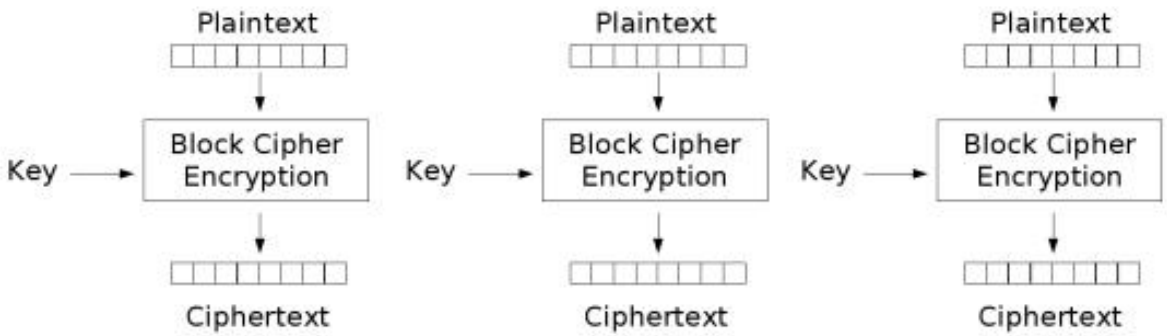
- 키를 이용하여 pseudorandom stream을 만들고 원래 데이터에 xor함
- c = m ^ G(k)
  - c : Encrypted text, m : plaintext, k : key, G : pseudorandom Generator
- G(k)값으로 k를 얻어낼 수 없어야 한다.
- OTP(One Time Pad)를 변형한 것
- 속도가 빠르고 구현이 간편하여 무선 통신 등에 이용
- RC4, A5/1, A5/2

대칭 키 - block cipher

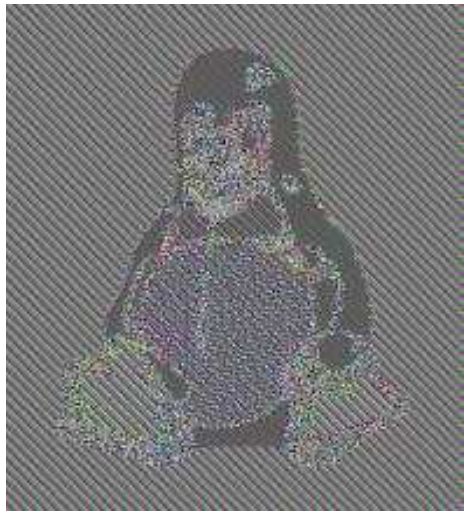
- Block 단위로 암호화 한다.
  - Block의 일부분이 변하면 전체가 변해야 함
- DES, 3DES, AES 등이 존재함
  - DES는 이제 brute force로 뚫린다.
- c = E(m,k), m = D(c,k)
- 블록보다 데이터가 길다면 운용 방식이 생김

Block cipher modes – ECB

- 키를 이용해 모든 블록을 각각 암호화



Electronic Codebook (ECB) mode encryption



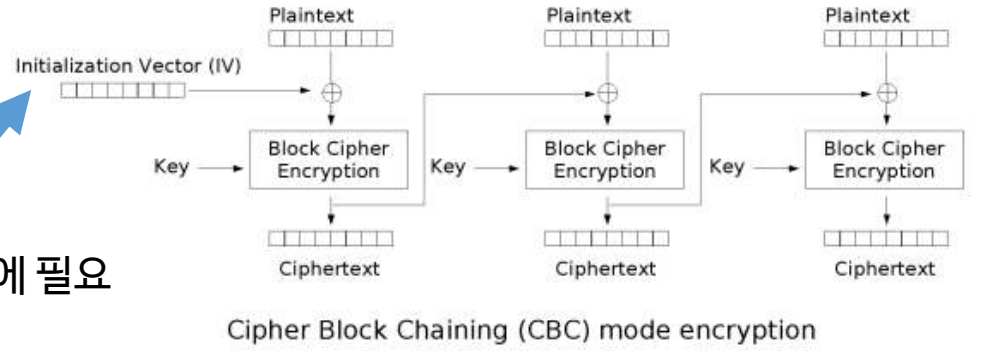
ECB 결과

- Block 내용이 같으면 결과가 같다

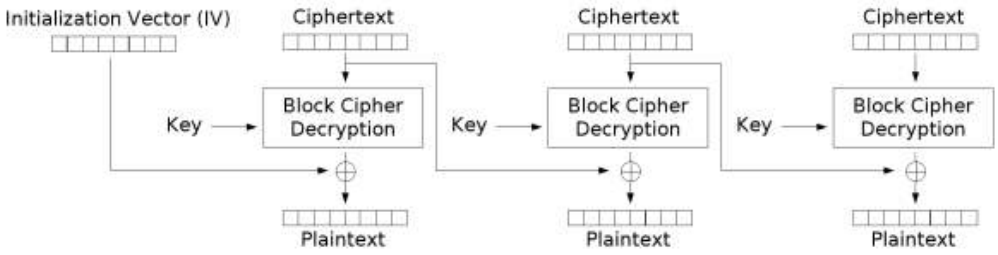
Block cipher modes – CBC

- 암호화 하기 전에 이전 암호화 결과와 XOR시킨다.

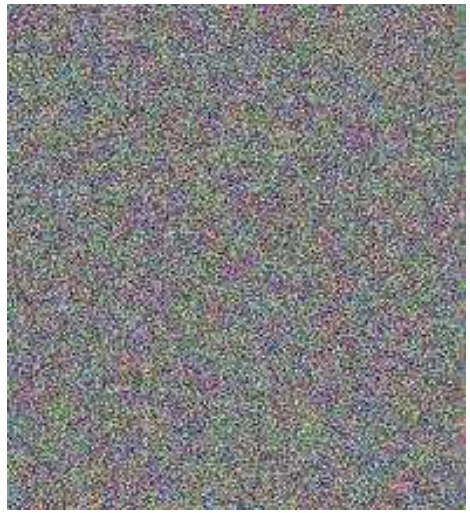
맨 처음에 필요



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption



851의 인수는?

851의 인수는?

- $\sqrt{851} \cong 29.2$
- 나눠본다. 2, 3, 5, 7, 11, …
- 답은 23,37

- 이렇게 작은 숫자조차도 구하기 힘들다 - NP 문제

비대칭 키 – RSA

- Rivest Shamir Adleman
- 소인수분해가 힘들다는 점을 이용한다.

RSA Key generation

- 큰 소수 2개를 구한다. (p,q)
- n=pq, z=(p-1)(q-1)
- z와 서로소인 e(e<n)를 구한다.
- ed mod z = 1 인 d를 구한다.
- (n,d)는 private key, (n,e)는 public key
- p,q를 놔두면 두 개의 키를 모두 만들 수 있으니 없앤다.

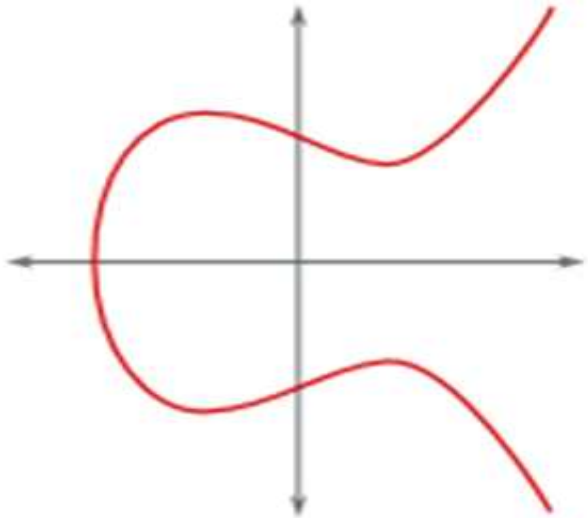
RSA Encryption/Decryption

- $c = m^e \ m \ od \ n$
- $m = c^d \ m \ od \ n$
- 왜?
- $(a \ m \ od \ n)^d = a^d m \ od \ n$
- $x^y \ m \ od \ n = \ x^{(y \ m \ od z)} \ m \ od \ n$
- 의 성질 이용

- $(m^e \ m \ od \ n)^d \ m \ od \ n = m^{ed} \ m \ od \ n = m$

비대칭 키 – ECC

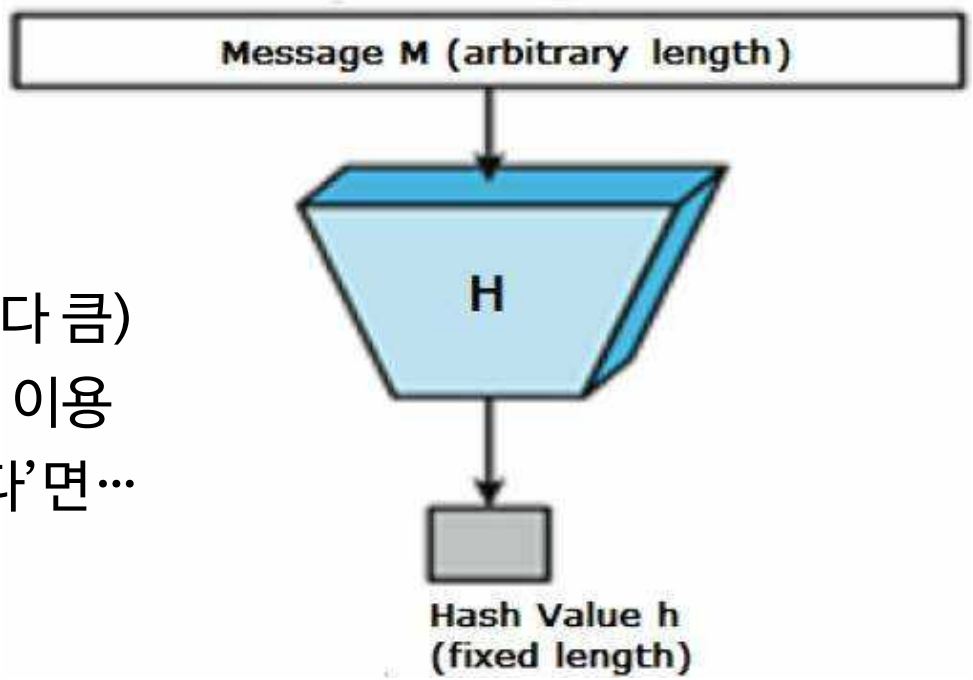
- 타원 곡선 암호 - Ecliptic Curve Cryptosystem
- 타원 곡선의 이산 로그를 찾는 것
- 짧은 키로도 RSA만큼 안전하다고 한다.



$y^2 = x^3 - x + 1$

해시

- 데이터를 특정한 길이의 데이터로 변환
- 다시 되돌릴 수는 없다. (정의역이 공역보다 큼)
- 비밀번호 보관, 데이터 위/변조 검증 등에 이용
- 만약 사이트에서 비밀번호를 '찾을 수 있다'면…



해시

- MD5, SHA 존재
- MD5 : collision attack이 가능하다.
- SHA1 : 올해 뚫렸다. ㅊㅊ
- https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html
- http://blog.alyac.co.kr/992 번역
- SHA-256, SHA-3를 쓰면 된다.

해시 – salt

- 해시는 brute force로 해결하는 수밖에 없다.
- 쉬운 비밀번호라면? 1q2w3e4r
- 이 것의 해시 값 자체를 저장해두고 그대로 이용할 수 있다.
- Salt는 비밀번호 뒤에 임의의 문자열을 넣어주는 것
- PBKDF2, BCRYPT, SCRYPT 방식이 있다.

TLS

- Transport Layer Security
- SSL(Secure Socket Layer)을 바탕으로 만든 보안 프로토콜
- 5계층 상으로는 application layer, 7계층 상으로는 session layer(transport layer 바로 위)
- 현재는 TLS v1.2를 이용한다. SSL이나 이전 버전은 쓰지 말자.

TLS handshake

- 서버와 클라이언트가 대칭 키를 만드는 과정

1. Client Hello
   - client에서는 사용 가능한 암호화 알고리즘 및 설정 등을 server에게 보낸다.
2. Server Hello
   - server는 자신의 인증서와 사용 가능한 알고리즘 등을 client에게 보낸다.
3. Client Key Exchange
   - 받은 인증서를 검증한 뒤, 대칭 키를 생성하고 이를 인증서의 공개키로 암호화해서 서버에게 보낸다.

TLS - 인증

- 주로 RSA Certificate를 이용한다.
- 인증서의 유효성을 보장해주는 발급자(issuer) 정보가 있다.
  - RSA signing을 이용해 보장

TLS - CA

- CA는 Self-signed certificate(발급자가 자신인 인증서)를 가지고 인증서를 발급해주는 기관을 의미한다.
- CA의 Self-signed certificate들은 브라우저 및 OS에 내장되어 있다.

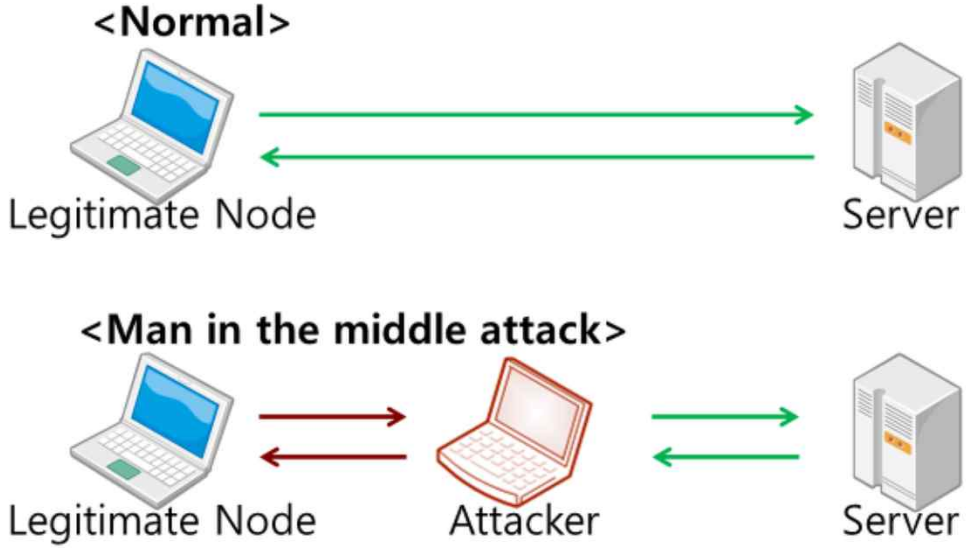
```
cert = input-certificate
while (true)
        if cert == root CA in Web Browser / OS
                PASS
        elseif cert is self-signed
                FAIL
        cert = cert's issuer-cert
```

TLS - certificate file types

- .crt: base64방식으로 인코딩한 인증서
- .cer: .crt와 같음 마이크로소프트에서 이용
- .key: base64방식으로 인코딩한 private key
- .pem: base64방식으로 인코딩한 private key 와 key 정보
  - Public key를 저장하는데 사용할 수도 있다.
- .pfx: certificate와 private key를 암호화
- .pem 또는 .key 파일이 유출되면 절대 안된다. (비밀키 정보이므로)
- 유출되었을 경우에는 인증서를 즉시 해지하고 재발급 받아야 한다.

중간자 공격(ManInTheMiddle Attack)

- 공격자가 중간에서 client / server와 연결하여 서로에게 상대방인 것처럼 보이게 한다.
- 모든 데이터를 볼 수 있으며, 적절하게 위/변조하여 보낼 수 있다.
- 이 공격이 수행되는 동안 공격자와 client는 서버의 인증서가 아닌 공격자의 인증서로 https 연결을 맺게 된다.
- 대부분의 브라우저는 인증서가 올바르지 않으면 연결을 차단한다.



SSL strip

- 공격자와 client간의 연결을 HTTP로 바꿔버린다.
- HTTPS 연결이 아니니 인증서로 인한 오류가 뜨지 않는다.

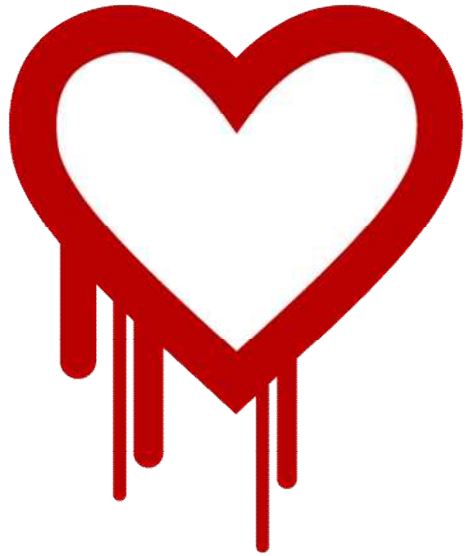
- HSTS : HTTP Strict Transport Security
  - Response header에 Strict-Transport-Security 정보를 보낸다.
  - 지정된 시간 동안 웹 브라우저는 해당 사이트에 무조건 https로 접속 시도한다.
  - MITM SSL Strip으로 인해 자동으로 http로 접속될 경우 페이지 로드를 중지한다.

OpenSSL

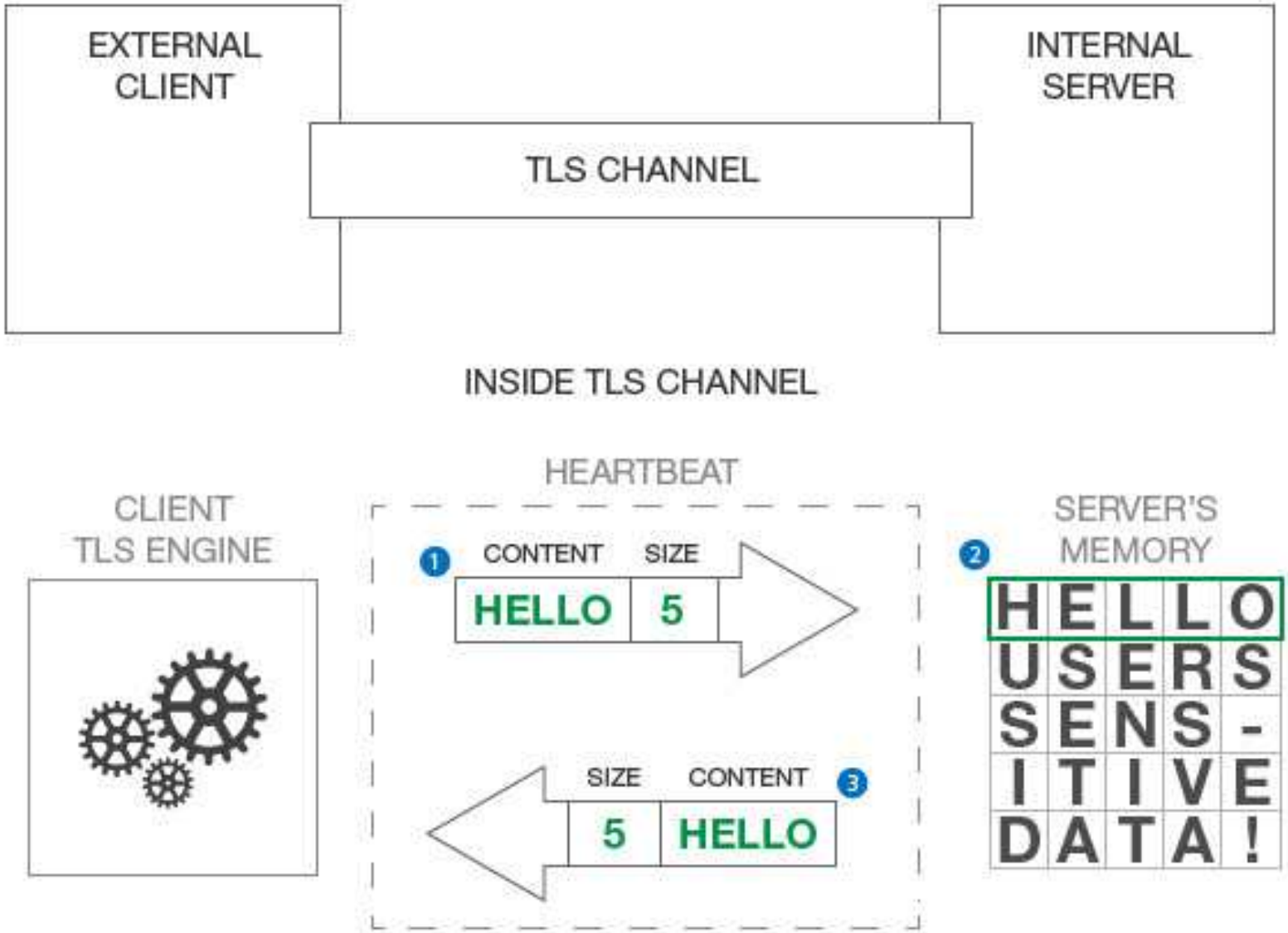
- SSL and TLS을 open source로 구현했다.
  - OpenSSL Library: 많은 암호 관련 함수들이 구현되어 있다.
    - AES, DES, RSA, ECC, Camellia, IDEA, MD5, SHA1, SHA2, DHKES …
  - 여러 open source 프로그램에서 이 library를 사용한다. (apache2 등)
- OpenSSL Toolkit: 암호화/복호화, 해시 구하기, 인증서 발급/검증 등 여러 작업을 할 수 있다.

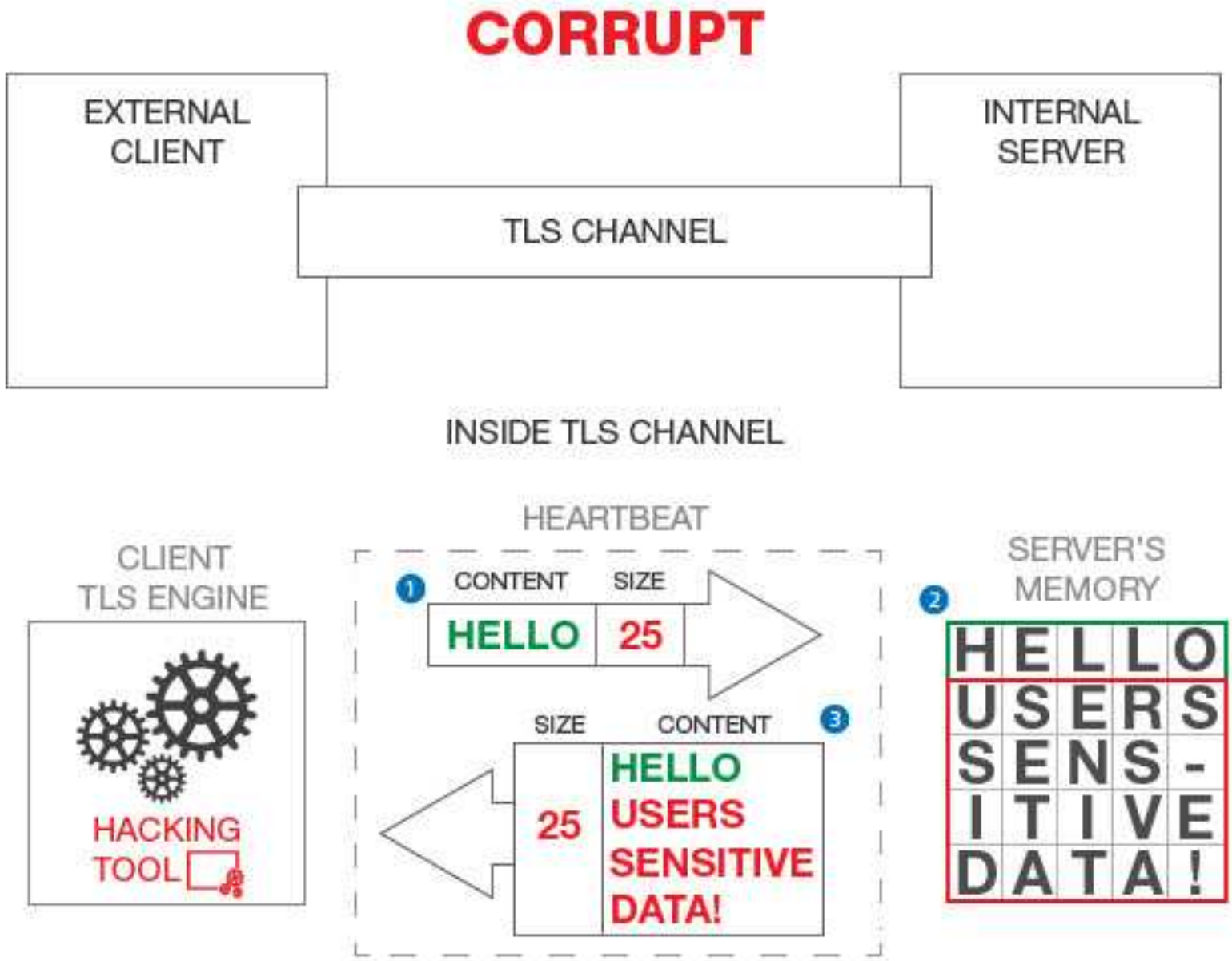
OpenSSL - heartbleed

- 2014년에 발견된 OpenSSL의 취약점
- OpenSSL의 heartbeat라는 통신 신호를 이용한 것이라 heartbleed 라고 부름
- https://xkcd.com/1354/ 에서 만화로 설명



CLEAN


EXTERNAL CLIENT
INTERNAL SERVER
TLS CHANNEL
INSIDE TLS CHANNEL
HEARTBEAT
CLIENT TLS ENGINE
1
CONTENT
SIZE
HELLO
5
SIZE
CONTENT
3
5
HELLO
2
SERVER'S MEMORY
H E L L O
U S E R S
S E N S -
I T I V E
D A T A !


CORRUPT
EXTERNAL CLIENT
INTERNAL SERVER
TLS CHANNEL
INSIDE TLS CHANNEL
HEARTBEAT
CLIENT TLS ENGINE
HACKING TOOL
1
CONTENT
SIZE
HELLO
25
SIZE
CONTENT
3
25
HELLO USERS SENSITIVE DATA!
SERVER'S MEMORY
2
HELLO
USERS
SENS-
ITIVE
DATA!

TLS - HTTPS

- HTTP over SSL
- TLS상에서 HTTP를 이용한다.
- HTTP 내용물이 암호화되어 있어 공격자가 볼 수 없다.
- 포트는 443
- 느리다.

HTTPS - letsencrypt

- 무료로 인증서를 준다.
- 90일간 이용 가능하다.
- 2017년 현재 apache와 nginx는 프로그램이 알아서 인증 절차+서버 설정을 진행해준다.
- 그 외의 경우는 알아서 인증 절차를 따르고, 본인이 쓰고 있는 서버 프로그램에 맞게 인증서를 등록하면 된다.
- 인증 절차는 간단한 편이다.
  - 그 사이트가 자신의 것인지 확인하기 위해 새로운 URL에 접속 가능하도록 만들어보라고 시킨다.

HTTPS - certbot

- 인증서 갱신을 대신 해준다.
- https://certbot.eff.org/ 에서 서버 소프트웨어와 OS를 선택
- 그에 맞는 설치 방법을 가르쳐준다.

TLS - SSH

- telnet : 원격으로 쉘을 띄우는 프로토콜. 암호화도 없고 중간자 공격도 못 막는다.
- SSH(Secure SHell) : 쉘을 안전하게 띄우는 프로토콜이다. 22번 포트를 이용한다.

공격/방어 - Brute force

- 그냥 가능한 모든 방법을 다 해보는 것
- 자전거 자물쇠를 0000부터 9999까지…
- 시간이 오래 걸리지만 계산속도가 빨라지면 언젠가 뚫을 수 있다.

- 방어법 : 시간이 오래 걸리게 한다.
  - 암호 길이를 길게 한다.
  - 가능한 key space를 늘린다(특수문자 사용 등)
  - 시간 당 대입 횟수를 제한한다.

공격/방어 - Dictionary Attack

- 많은 사용자들이 본인의 이름, 생일 등의 정보나 쉬운 단어로 비밀번호를 만든다.
  - inseung1996 fuxxingpassword apple1234 qlalfqjsgh …
- 단어를 잘 조합해서 비밀번호로 넣어보는 공격
- 방어법 : 비밀번호를 단어, 내 정보 등으로 만들지 않는다.

공격/방어 - Fail2ban

- Linux 사용자 로그인에서, 비밀번호를 n번 이상 틀리면 m초 동안 로그인할 수 없게 만드는 프로그램이다.

- 설치 및 적용법 :
- # apt-get install fail2ban
- # cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
- # vi /etc/fail2ban/jail.local

공격/방어 - log system

- 기본적으로 /var/log/ 에 들어있다. (Debian/Ubuntu 기준)
  - 시스템 로그: syslog
  - 보안 로그: auth.log
  - 크론 로그: cron
  - 부팅 로그: boot.log
  - 웹 로그: apache2/
- 로그 파일 이름은 배포판마다 약간씩 달라진다.

File Permission - setuid, setgid

- Set User / Group ID upon execution
- /etc/passwd는 root만 쓰기 권한이 있다.
  - 하지만 모든 유저가 passwd를 실행함
- setuid가 설정되어 있는 파일은 실행 시 소유자의 권한으로 실행된다.
- setgid가 설정되어 있는 파일은 실행 시 소유 그룹의 권한으로 실행된다.

File Permission - sticky bit

- /tmp, /var/tmp은 기본적으로 permission 777이다.
- 다른 사람이 사용중인 파일을 지워버리면 서비스에 장애가 발생할 수 있다.
- Sticky Bit: 이 bit가 설정된 directory 안에서는 파일의 소유자만이 삭제 가능 & directory의 소유자만이 삭제 가능
- Ex) chmod 1777 somedirectory
  - ls -l
  - drwxrwxrwx**t**

Firewall (iptables)

- 3가지 chain이 존재하여 규칙을 수행한다.
  - INPUT: 들어오는 패킷 / OUTPUT: 나가는 패킷 / FORWARD: 통과하는 패킷
- 여러가지 조건으로 필터링할 수 있다.
  - source, destination, state, protocol, …
- 패킷에 대하여 다양한 동작을 할 수 있다.
  - accept, drop(그냥 버림), reject(버리면서 알려줌)

Firewall (iptables)

- 네트워크 연결 상태에 따라서 필터링할 수 있다.
  - NEW: 새로운 연결을 요청하는 패킷
  - ESTABLISHED: 기존 연결의 일부인 패킷
  - RELATED: 기존 연결에 속하지만 새로운 연결을 요청하는 패킷
    - ex) 접속 포트가 20인 FTP가 65535를 이용하여 전송하고 싶어한다.
  - INVALID: 나머지

Firewall (iptables)

- -A: 규칙 추가, -D: 규칙 삭제, -L: 규칙 출력, -P: 기본 정책 변경
- 모든 곳에서 eth0으로 들어오는 ssh 연결을 허용한다
  - iptables -A INPUT -i eth0 -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
  - iptables -A OUTPUT -o eth0 -p tcp --sport 22 -m state --state ESTABLISHED - j ACCEPT
- 모든 곳에서 eth0으로 들어오는 mail 연결을 허용한다.
  - iptables-A INPUT -ieth0 -p tcp--dport25 -m state --state NEW,ESTABLISHED -j ACCEPT
  - iptables-A OUTPUT -o eth0 -p tcp--sport 25 -m state --state ESTABLISHED -j ACCEPT

Firewall (UFW)

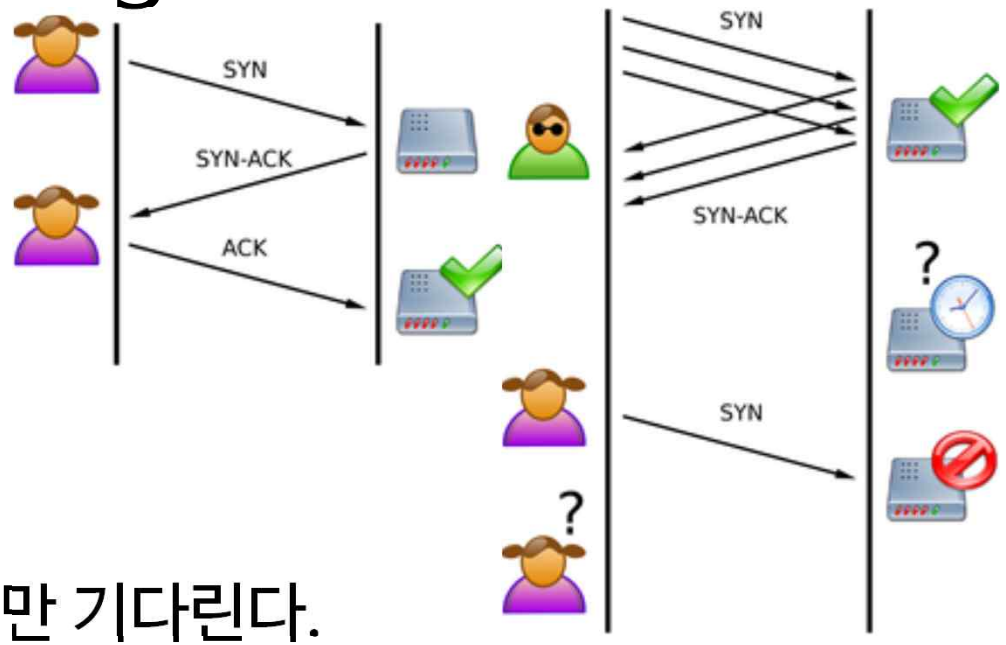
- Uncomplicated FireWall
- 설치: # apt-get install ufw
- 켜고 끄기: # ufw [enable | disable]
- 상태 보기: # ufw status
- Rule 추가하기: ufw [allow | deny | reject] 〈rule syntax〉
- 〈rule syntax〉 : 〈port no〉, 〈port no〉/〈protocol〉, service_name from 〈from_ip_range〉 to 〈to_ip_range〉
- Rule 삭제하기: ufw delete [allow | deny | reject] 〈rule syntax〉

공격/방어 - DoS/DDoS

- (Distributed) Deniel of Service
- 한정된 네트워크 자원을 모두 소모시켜 정상적인 사용자가 서비스에 접근하지 못하도록 하는 공격이다.

공격/방어 - SYN Flooding

- DoS 공격의 대표적인 방법



- TCP연결의 SYN만 잔뜩 보낸다.
- 서버는 SYN에 대한 답을 보내고 ACK만 기다린다.
- 서버가 연결을 기다리느라 다른 사용자가 연결하지 못한다.

공격/방어 - Web based attack

- 웹 페이지의 입력 공간에 공격 문자열을 삽입한다.
- GET / POST 등을 사용하여 통신되는 데이터를 변조한다.
- Ex) SQL injection, XSS, CSRF

공격/방어 - SQL injection

- 서버에서 본인이 원하는 SQL 구문을 실행하는 공격 방법
- SELECT id FROM user WHERE id='{id}' AND pw='{pw}'
  - pw에 'OR 1='1을 넣으면
  - SELECT id FROM user WHERE id='his' AND pw='' OR 1='1'
- 방어법: 기본적으로 입력을 필터링해야 한다.
  - Django의 경우 대부분 raw SQL을 프로그래머가 작성하지 않으므로 안전하다.
  - PHP의 경우 각별한 주의가 필요하다. mysql_real_escape_string을 이용한다.

공격/방어 – XSS

- Cross Site Scripting
- 사용자가 특정 페이지를 신뢰하여 이에 접속하면 공격자가 원하는 script를 사용자가 실행하도록 하는 공격 방법이다.

1. 공격자는 악의적인 script (ex: 쿠키 훔치기)를 만든다.
2. 공격 대상이 될 웹사이트에 본인이 만든 악의적인 script를 삽입한다.
   - (ex) iframe, object, div, script 태그 등을 게시글로 작성한다
3. 사용자가 해당 페이지를 열어보면 script가 실행된다.

- 방어법: 특정 문자를 필터링하면 된다.
  - Ex) 〈는 <, 〉는 > 로 escape한다.

공격/방어 - CSRF

- Cross Site Request Forgery
- 해당 웹 사이트가 사용자의 웹 브라우저를 신뢰하여, 사용자가 요청 하지 않은 명령을 공격자가 보내도록 하는 공격 방법이다.
- Ex) http://bank.com/withdraw?amount=999999999&to=his
  - 만약 로그인 되어있는 상태로 저 링크를 누르면 his에게 돈을 보내게 될 수도 있다.
  - 〈img src="http://bank.com/withdraw?amount=999999999&to=his"〉가 있는 사이트를 들어가기만 해도…

- 방어법: 서버 쪽에서 random한 token이 포함된 페이지를 사용자에게 보내고, 사용자에게서 요청을 받을 때는 token이 올바른지 확인한다.
  - Ex) django에서 views.py의 `@csrf_token`(csrfmiddlewaretoken)

Reference

- https://developer.mozilla.org/en-US/docs/Archive/Security/Encryption_and_Decryption#Symmetric-Key_Encryption
- https://ko.wikipedia.org/wiki/%EC%8A%A4%ED%8A%B8%EB%A6%BC_%EC%95%94%ED%98%B8
- https://ko.wikipedia.org/wiki/%EB%B8%94%EB%A1%9D_%EC%95%94%ED%98%B8_%EC%9A%B4%EC%9A%A9_%EB%B0%A9%EC%8B%9D
- https://www.tutorialspoint.com/cryptography/cryptography_hash_functions.htm
- http://starplatina.tistory.com/entry/%EB%B9%84%EB%B0%80%EB%B2%88%ED%98%B8-%ED%95%B4%EC%8B%9C%EC%97%90-%EC%86%8C%EA%B8%88%EC%B9%98%EA%B8%B0-%EB%B0%94%EB%A5%B4%EA%B2%8C-%EC%93%B0%EA%B8%B0
- http://blog.yoko.so/entry/%ED%95%98%ED%8A%B8%EB%B8%94%EB%A6%AC%EB%93%9C-%EA%B3%B5%EA%B2%A9%EA%B3%BC-%EB%B0%A9%EC%96%B4
- https://blog.outsider.ne.kr/1178