Security

SPARCS WHEEL SEMINAR 2015-07-21 SAMJO

주의 – 앞부분은 상당히 재미없음

Index

- Cryptography
  - Cipher (Symmetric / Asymmetric)
  - Hashing
  - Authentication
  - Key Exchange Protocol (Diffie-Hellman)
- Transmission Layer Security (TLS)
- Attack and Defense
  - Brute Force, Log, File Permission
  - Network – Firewall, SYN Flooding
  - Web – SQL Injection, XSS
  - Physical

Security – Goals!

- Confidentiality: 비밀 정보는 비밀로 간직해야 함!
  - Cipher, TLS/SSL, File Permission
- Integrity: 정보는 신뢰 할 수 있어야 함!
  - Hash Function, TLS/SSL, Certificate
- Availability: 서비스는 항상 접근 가능해야 함!
  - DoS / Ddos Defense
- Authenticate: 저 사람이 진짜 저 사람이 맞아야 함!
  - Password-based, Certificate

Security – Practical Goal!

- root 권한을 정해진 사람만 가질 수 있도록 해야 함
  - 외부인이 root 권한을 가지게 하면 안됨
  - 내부인이 권한 상승 취약점 등을 통해 root 권한을 가지게 하면 안됨
  - root 권한을 가진 사람이라도 이상한 행동을 하도록 하면 안됨
    - ex: rm -rf / --no-preserve-root

- c.f root의 초기 비밀번호는 설정되어 있지 않음
  - = 설정하기 전까지 root로 로그인할 수 없음

Cryptography

- “The practice and study of techniques for secure communication in the presence of third parties. It is about constructing and analyzing protocols that block adversaries.”

- Topics: **Cryptosystem**, **Hash Function**, Message Authentication Code, Random Number Generation, **Key Exchange Protocol**, **Authentication**, Steganography, Zero-Knowledge Proof, Secret Sharing, Quantum, ...

Cipher

- “an algorithm for performing encryption or decryption”
  - Encryption: “process of encoding messages or information in such a way that only authorized parties can read it” – Wikipedia
- 특정인만 읽을 수 있도록 데이터를 변환하는데 사용되는 알고리즘
- Mainly 1. Key Generation / 2. Encryption / 3. Decryption

Terms & Definition

- Plaintext (평문): 암호화하고 싶은 정보
- Ciphertext (암호문): 암호화 된 정보
- Key (키): 암호화할 때 필요한 부가적인 정보
  - A piece of information that determines the functional output of a cryptographic algorithm or cipher
  - 이게 없으면 유의미한 암호화 알고리즘을 만들 수 없음

Mathematically, a cryptosystem or encryption scheme can be defined as a tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ with the following properties.

1. $\mathcal{P}$ is a set called the "plaintext space". Its elements are called plaintexts.
2. $\mathcal{C}$ is a set called the "ciphertext space". Its elements are called ciphertexts.
3. $\mathcal{K}$ is a set called the "key space". Its elements are called keys.
4. $\mathcal{E} = \{E_k : k \in \mathcal{K}\}$ is a set of functions $E_k : \mathcal{P} \to \mathcal{C}$. Its elements are called "encryption functions".
5. $\mathcal{D} = \{D_k : k \in \mathcal{K}\}$ is a set of functions $D_k : \mathcal{C} \to \mathcal{P}$. Its elements are called "decryption functions".

For each $e \in \mathcal{K}$, there is $d \in \mathcal{K}$ such that $D_d(E_e(p)) = p$ for all $p \in \mathcal{P}$.[2]

Symmetric vs Asymmetric

- Symmetric-Key Cipher(대칭키 암호화 알고리즘)
  - Encryption, Decryption시 동일한 key를 사용

- Asymmetric-Key Cipher(비대칭키 암호화 알고리즘)
  - Public Key(공개키) – Private Key(비밀키) 쌍을 생성
  - Encryption시 public key를, decryption시 private key 사용

- 빠른 속도: symmetric > asymmetric
- 빠른 이해: symmetric > asymmetric

Why Asymmetric?

- 잠시 만화 보고 갑니다.
  - https://www.facebook.com/media/set/?set=oa.9939767539761 94&type=1
- Key Distribution Problem
  - N명이 Symmetric Cipher: O(N^2)
  - N명이 Asymmetric Cipher: O(N)
- Cryptographic Signing (서명)
  - 알고리즘의 특징을 사용하여 “서명” 가능
  - Authentication 때 자세히 설명

Symmetric Cryptosystem

For each $e \in \mathcal{K}$, there is $d \in \mathcal{K}$ such that $D_d(E_e(p)) = p$ for all $p \in \mathcal{P}$.

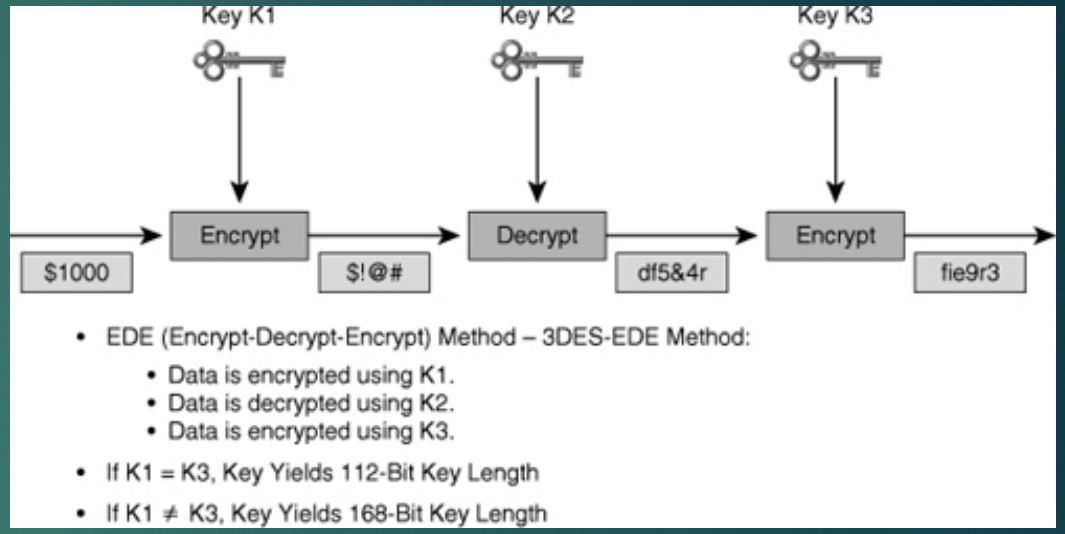
For each $e \in \mathcal{K}$, ~~there is $d \in \mathcal{K}$ such that~~ $D_{\underline{d}}(E_e(p)) = p$ for all $p \in \mathcal{P}$.
$\underline{e}$

- Stream Cipher: “plaintext digits are combined with a pseudorandom cipher digit stream (key stream)”
  - In practical, digit: 1 bit or byte, combine: XOR
  - Loose form of OTP (one time pad: ultimate, unbreakable cipher)

  부적절한 Stream Cipher의 사용은 심각한 암호학적 문제를 발생시킬 수 있음!
- Block Cipher: “algorithm operating on fixed-length groups of bits, called blocks, with an unvarying transformation”
  - 이 분류의 암호화 알고리즘이 실제 많이 사용됨

Block Cipher – DES, 3DES

- Data Encryption Standard (DES)
  - 1975, IBM / 64bit block / 56bit key
  - NIST에 의해 연방정부 표준으로 승인됨 (46-3)
  - **Brute force attack is possible!!**



- Triple-DES (3DES)
  - DES가 안전하지 않으면 3번 중첩해서 쓰면 됨
  - 그러나, 속도가 느리고 DES가 가지고 있는 다른 문제(weak key의 존재, differential cryptanalysis에 취약함)를 여전히 가지고 있음

Block Cipher – AES, SEED and Others

- Advanced Encryption Standard (AES, Rijndael)
  - 1998, Vincent Rijmen, Joan Daemen / 128bit block / 128, 192, 256bit key
  - NIST에 의해 연방정부 표준으로 승인됨 (197)
  - NSA에서 최고 등급 기밀을 저장할 때 사용할 수 있도록 승인됨
  - ISO 표준 / S/MIME, TLS/SSL, IPSec, X509 등에서 사용
- SEED
  - KISA(Korea) / 128bit block / 128bit, 256bit key
  - ISO 표준 / S/MIME, TLS/SSL, IPSec, X509 등에서 사용
- Other: ARIA, Twofish, Camellia, CAST-128, ...

Asymmetric Cipher

For each $e \in \mathcal{K}$, there is $d \in \mathcal{K}$ such that $D_d(E_e(p)) = p$ for all $p \in \mathcal{P}$.

- e (public key), E (enc. fun.)는 공개되어 있음
  - 즉, 다음의 조건을 만족하는 함수를 찾아야 함
  - 1. d (private key)가 있으면 E의 역함수를 만들 수 있어야 함
  - 2. d가 없으면 E의 역함수를 만들기 거의 불가능해야 함
  - 3. e, E로 부터 d를 찾기는 거의 불가능해야 함

- Trapdoor One-Way Function
  - 수학적인 난제 (주로 NP-Hard)를 사용함
  - Integer Factorizing Problem => RSA
  - Discrete Logarithm Problem => ECC, DH

RSA (Rivest, Shamir, Adleman)

- Based on Integer Factorization Problem
  - Given a large integer n, factorize it!

- 1. Key Generation
  - 소수 p, q를 선택; n=pq; e는 phi(n)과 서로소인 임의의 숫자; d는 phi(n)에 대한 modular inverse
  - Public key: (n, e), Private Key: (p, q, d)

- 2. Encryption (plaintext: m): m^e mod n

- 3. Decryption (ciphertext: c): c^d mod n

RSA bits

- RSA 512bit, 1024bit, 2048bit, 4096bit 등이 현재 사용

- 여기서 bit란, n의 크기를 나타냄
  - ex: RSA 1024bit => 2^1023 <= n < 2^1024

- **RSA 512 bit: BROKEN; NEVER USE**
  - i.e p*q=n이 되는 p와 q를 구할 수 있다!
- **RSA 1024 bit: THEORETICAL ATTACK FOUND**
- RSA 2048 bit: RECOMMENDED

Does it really works?

- 잘 됩니다. “Wheel 세미나”이니 자세한 증명은 생략합니다.
- 자세한 증명을 알고 싶다면,
  - CS448 정보보호개론 수업을 들으시거나,
  - https://en.wikipedia.org/wiki/RSA_(cryptosystem)#Proofs_of_correctness

Other Asymmetric Cipher

- Diffie-Hellman Key Exchange Protocol
  - 뒤에서 자세히 설명
- ECC (Elliptic Curve Cryptosystem)
  - Based on discrete logarithm problem
  - Trendy, RSA보다 키가 상대적으로 짧음
- ElGamal: discrete logarithm problem
- Paillier: additive homomorphic
  - E(a) + E(b) = E(a + b)

Hashing (Hash Function)

- “any function that can be used to map digital data of arbitrary size to digital data of fixed size”

- Example: for any x, x mod 543082734

- Meaningful Hash
  - Preimage resistance: given y, infeasible to find x s.t y=h(x)
  - Weak collision resistance: given x, infeasible to find x’ s.t h(x)=h(x’)
  - Strong collision resistance: infeasible to find x, x’ s.t h(x)=h(x’)

Hash Algorithms

- MD5 (Message Digest 5)
  - Rivest / 1992 / 128bit
  - **NOT RESISTANCE TO COLLISION ATTACK; DO NOT USE**
- SHA-1 (Simple Hash Algorithm 1)
  - NSA / 1995 / 160bit
  - Theoretical attack was found. No actual collision attack.
- Use SHA-256, SHA-512, SHA-3 (Keccak)

Hash Application

- 비밀번호 보관
  - “preimage resistance”
- 파일, 데이터 위/변조 검증
  - “weak collision resistance”
- 파일, 데이터 식별자
  - “strong collision resistance”

Authentication

냥이냥냥

- “act of confirming the truth of an attribute of a single piece of data or entity”
- Example: 이 PPT는 samjo가 만든 것입니다. 왜냐하면 ...
- RSA에서 encryption부분과 decryption 부분을 바꿔보자!
  - decryption -> signing
  - encryption -> verifying

Signing with RSA

- 준비물: RSA key (n, e, d), Hash function h
- 1. Signing (text: m): (m, h(m)^d mod n)
- 2. Verifying (m, s): h(m) =? s^e mod n

Other Authentication Methods

- ElGamal Signature Scheme
- DSA (Digital Signature Algorithm)
  - NIST에 의해 연방정부 표준으로 승인됨 (186)
  - ElGamal Signature Scheme의 Varient
- ECDSA (Elliptic Curve -)
- Schnorr Signature
  - Based on discrete logarithm problem.

Key Exchange Protocol

- Alice가 Bob과 안전하게 채팅을 하고 싶어한다.
- 그냥 메시지를 보내면 나쁜 samjo가 중간에서 메시지를 도청한다.
- AES 알고리즘을 사용하고 비밀키 K를 이용하여 암호화하자!

- **근데 K는 어떻게 안전하게 보내지?**

Discrete Logarithm Problem

- Question: given y, a, find x s.t y = a^x
  - Solution: x = log_a y; simple and easy

- Question: given y, a, p, find x s.t y = a^x mod p
  - Solution: ...????????
  - “Discrete Logarithm Problem”

Diffie-Hellman KES

- 1. Pick G = Z*^p, g s.t g is generator of G
- 2. Alice picks a random number “a”
  - And sends g^a mod p to Bob
- 3. Bob picks a random number “b”
  - And sends g^b mod p to Alice
- 4. Alice computes (g^b)^a mod p
- 5. Bob computes (g^a)^b mod p

Diffie-Hellman KES - Safe!

- samjo는 g^a mod p, g^b mod p를 알 수 있다.
- 하지만, 이 정보로부터 a, b를 구할 수 없다.
  - By discrete logarithm problem
- 또한, 이 정보로부터 g^ab mod p를 구할 수 없다.
  - By Diffie-Hellman assumption

- Usage: **SSH(secure shell)**, TLS/SSL, …

TLS (Transport Layer Security)

- “cryptographic protocols designed to provide communications security over a computer network”
  - On OSI Layer 5 – Session layer

- SSL (Secure Socket Layer) 1.0, 2.0, 3.0 by Netscape
- TLS 1.0 (1999-01) defined in RFC 2246, upgrade from SSL 3.0
- TLS 1.1 (2006-04) defined in RFC 4346
- TLS 1.2 (2008-08) defined in RFC 5246

- **SSL 1.0, 2.0: NEVER USE**
- **SSL 3.0: DO NOT USE (POODLE attack)**

TLS Overview

- Client와 server는 같은 키를 생성 (handshake)
- 생성된 대칭키로 데이터를 암호화하여 주고 받음 (ex: AES)

- 1. Client Hello
  - Client에서는 사용 가능한 암호화 알고리즘 및 설정 등을 server에게 보냄
- 2. Server Hello
  - Server는 자신의 **인증서**와 사용 가능한 알고리즘 등을 client에게 보냄
- 3. Client Key Exchange
  - 받은 인증서를 **검증**한 뒤, 대칭키를 생성하고 이를 인증서의 공개키로 암호화해서 서버에게 보냄

Certificate

- “an electronic document used to prove ownership of a public key. (X.509)”

  = public key with owner and issuer information

- 거의 대부분이 RSA certificate
- 인증서의 유효성을 보장해주는 발급자(issuer) 정보가 있음
  - 1. 어떻게 “보장”해 줄 수 있음? => RSA Signing
  - 2. 발급자는 어떻게 믿지? => Root CA

Certificate File

- .crt: base64 encoding of certificate
- .cer: same as .crt, used by Microsoft
- .key: base64 encoding of private key
- .pem: base64 encoding of private key and key's info
  - Public key를 저장하는데 사용할 수도 있다.
- .pfx: certificate와 private key를 암호화하여 저장

- .pem 또는 .key 파일이 유출되면 절대 안됨
- 유출되었을 경우에는 인증서를 즉시 해지하고 재발급 받도록 하자!

PKI (Public Key Infrastructure)

- “A set of hardware, software, people, policies, and procedures need to create, manage, distribute, use, store and revoke digital certificates”
- 인증서 발급 정책 등을 포함한 다양한 요소가 포함된 개념
- 이 세미나에서는 어떻게 인증서를 validate하는지만 설명함

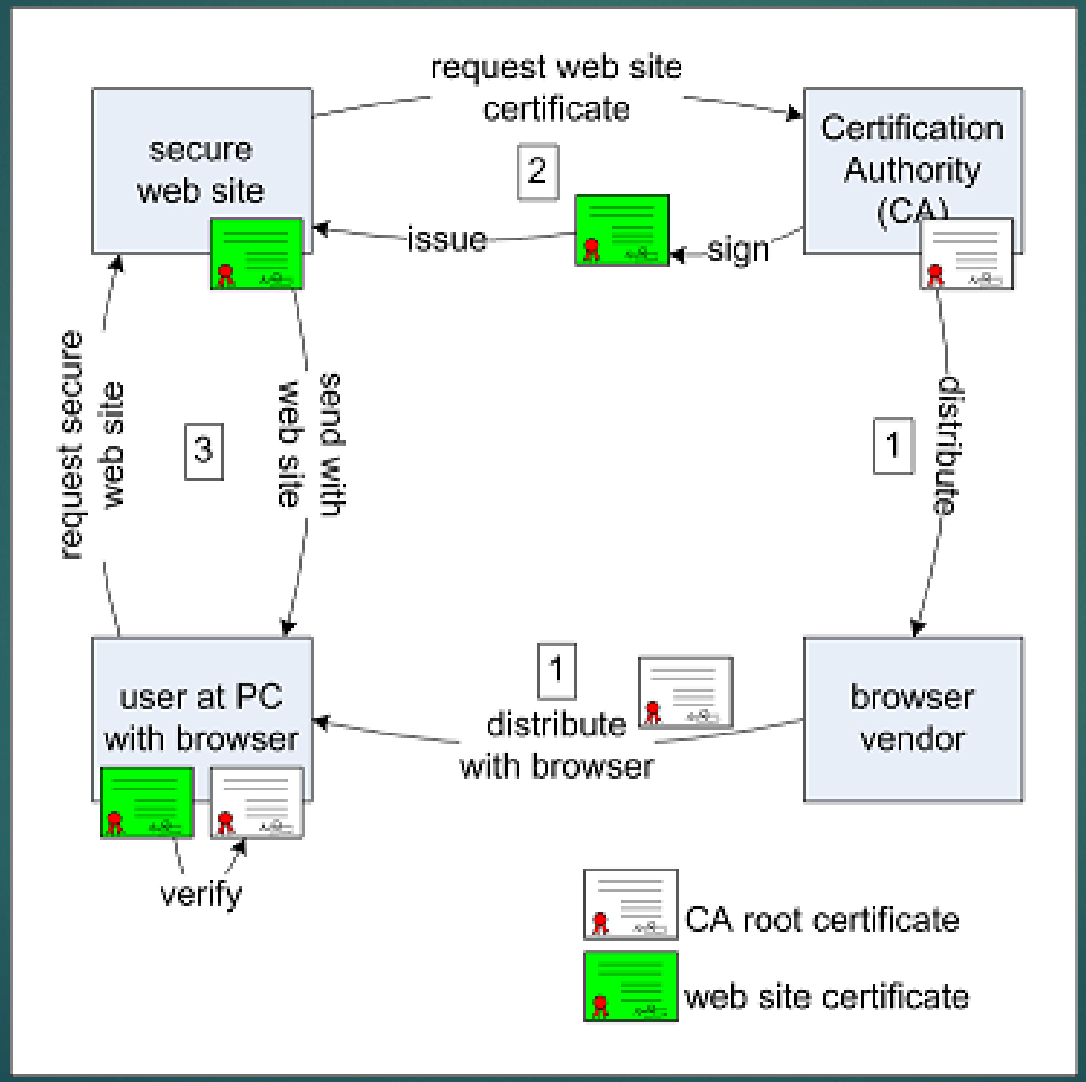
Root Certificate Authority (CA)

- Self-signed certificate를 가지고 인증서를 발급해 주는 기관
  - Self-signed certificate: issuer이 자신인 인증서
  - ex: Comodo, Symantec, GoDaddy, GlobalSign, ...
- 오프라인에서 다른 회사의 감사를 받음
- Root CA의 Self-signed certificate들은 브라우저 및 OS에 내장

How to Validate Certificate?

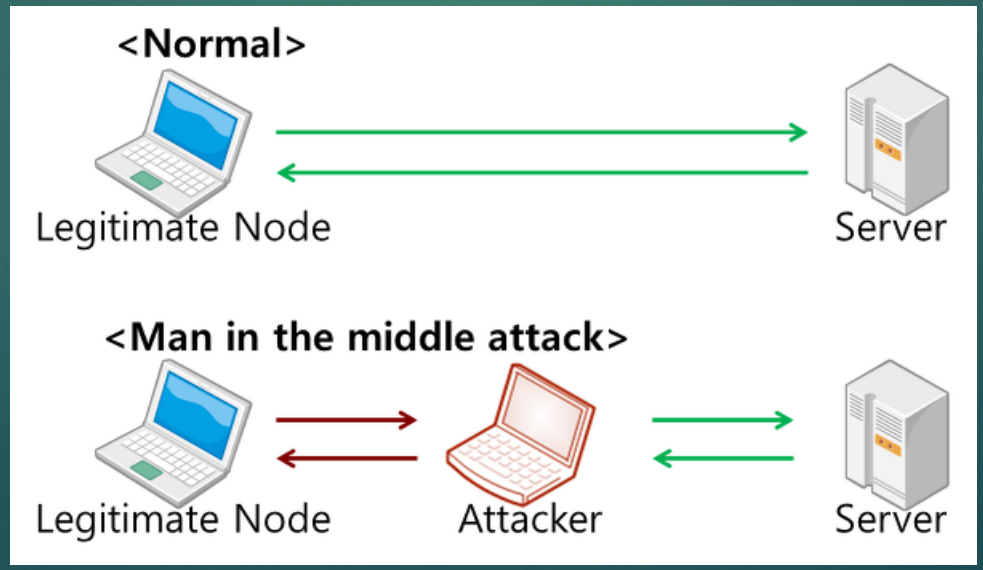
```
cert = input-certificate
while (true)
    if cert == root CA in Web Browser / OS
        PASS
    elseif cert is self-signed
        FAIL
    cert = cert's issuer-cert
```

TLS/SSL Summary



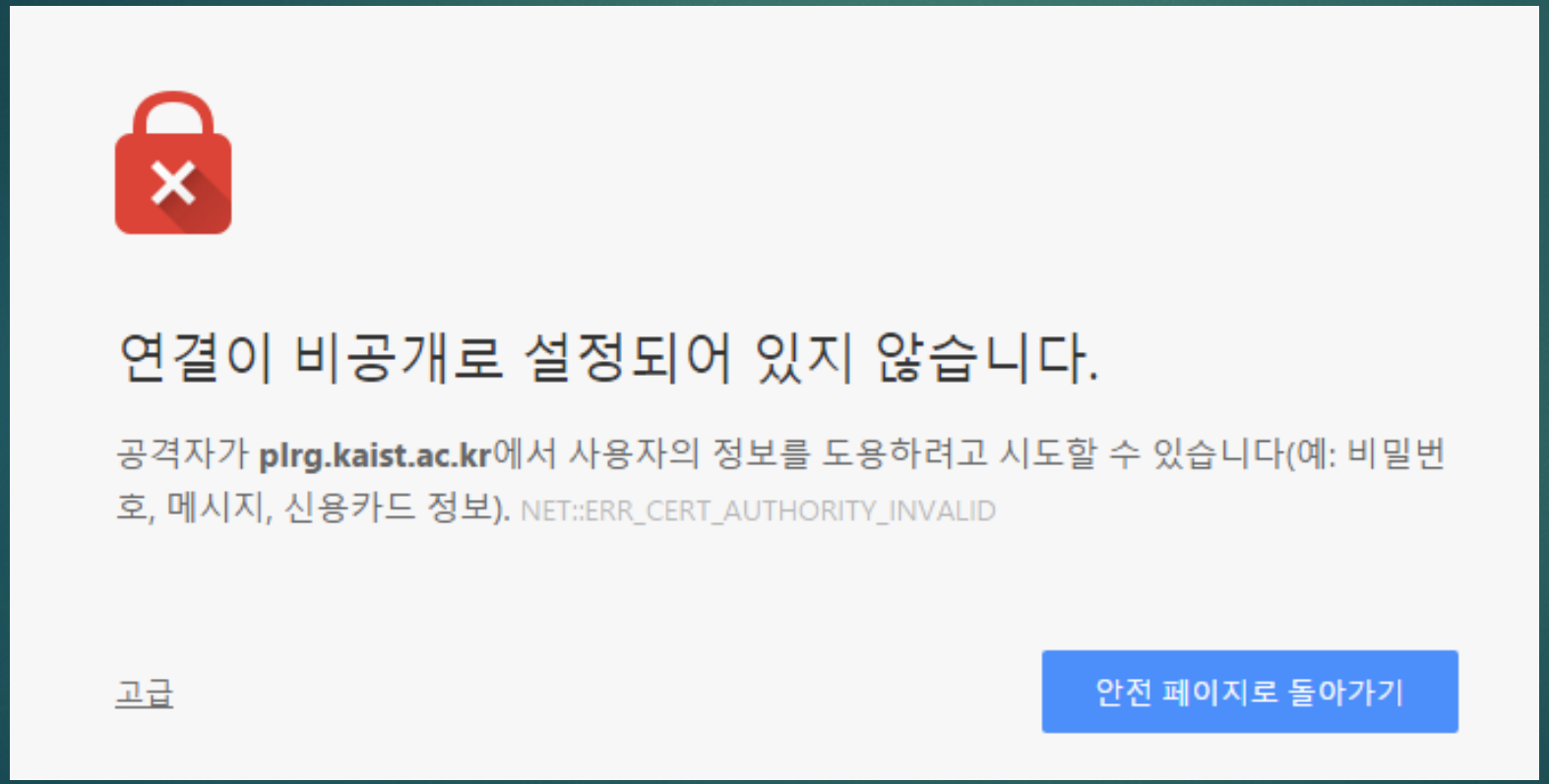
Why need to validate?

- MITM Attack (Man-In-The-Middle Attack)
  - 공격자가 중간에서 client와 연결 / server와 연결하여 서로에게 상대방인것처럼 보이게 하는 공격 방법
  - 당연히 모든 데이터를 볼 수 있으며, 적절하게 위/변조하여 보낼 수 있음
  - 이 공격이 수행되는 동안 attacker와 client는 서버의 인증서가 아닌 공격자의 인증서로 https 연결을 맺게 됨



CA Invalid Error

- 거의 모든 웹 브라우저에서는 MITM 공격을 방지하기 위해, valid 하지 않은 인증서로 SSL연결을 시도할 경우 진행을 중단



SSL Strip

- 인증서를 위조하였더니 client에게 잘못된 인증서라는 경고창이 뜸
- 그러면 그냥 attacker와 client사이에 http 연결을 사용하자!
  - 사용자는 http 통신이 이루어지는지, https 통신이 이루어지는지 무관심함

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

OpenSSL

- SSL and TLS의 open source 구현
- OpenSSL Library: 많은 암호 관련 함수들이 구현되어 있음
  - AES, DES, RSA, ECC, Camellia, IDEA, MD5, SHA1, SHA2, DHKES …
  - 여러 open source 프로그램에서 이 library를 사용함 (apache2 등)
- OpenSSL Toolkit: 암호화 / 복호화, 해시 구하기, 인증서 발급 / 검증 등 여러 작업을 할 수 있음
  - **openssl --version**

Heartbleed

- 2014년에 발견된 OpenSSL의 구현 취약점
  - http://heartbleed.com/
- Heartbeat extension (RFC6520) 구현에서 boundary check를 하지 않아 서버의 메모리 내용이 유출되는 취약점
- 자세한 원리: 만화를 봅시다!
  - https://xkcd.com/1354/

HTTP & HTTPS

- HTTP: HyperText Transfer Protocol
  - 기억하고 계시나요? (ref Web, Apahce & Nginx)

- HTTPS: HTTP over SSL
  - HTTP 데이터를 암호화하여 통신함
  - 공격자가 데이터를 볼 수 없음
  - URL: https:// (NOT http://)
  - Port: 443 (NOT 80)
  - **“매우 중요한 단점: 느리다”**

Tutorial

- Tutorial 목표: openssl을 사용하여 self-signed certificate를 만들어보고, apache에 적용시켜 https 서버를 만들어보자!
- Tutorial 과정:
  - 1. **openssl genrsa -des3 -passout pass:x -out server.pass.key 2048**
  - 2. **openssl rsa -passin pass:x -in server.pass.key -out server.key**
  - 3. **rm server.pass.key**
  - 4. **openssl req -new -key server.key -out server.csr**
    - Common Name에는 실습 서버 ip 또는 도메인을 넣도록 하자
  - 5. **openssl x509 -req -days 365 -in server.csr -signkey server.key -out site.crt**
  - 6. **vi /etc/apache2/sites-available/default-ssl.conf**
    - SSLCertificateFile /home/samjo/site.crt
    - SSLCertificateKeyFile /home/samjo/server.key
  - 7. **a2enmod ssl; a2ensite default-ssl; service apache2 restart;**

Telnet & SSH

- Telnet: Session layer protocol used to provide a virtual terminal connection (RFC 15, RFC 854)
  - NO Encryption; NO MITM Prevention; DO NOT USE;

- SSH (Secure SHell): shell을 안전하게 띄울 수 있도록 하는 protocol
  - Port: 22

FTP, FTPS & SFTP

- FTP(File Transfer Protocol)
  - NO Encryption; NOT RECOMMENDED
- FTPS(FTP on SSL)
  - SSL연결을 맺고, 그 위에서 FTP 통신을 함
- SFTP(Secure FTP)
  - SSH에서 터미널 접속 뿐만 아니라 파일 관련 작업을 할 수 있도록 만들어 놓은 일종의 extension
  - FTP와 protocol 자체가 다르므로, 명령어 형식도 다름

Attack and Defense

- 창과 방패의 싸움

- Brute Force
- Log System
- File Permission (setuid, setgid, sticky bit)
- Firewall (iptables)
- SYN Flooding / DDos Attack
- SQL Injection, XSS, CSRF
- Physical Security

Brute Force

- Key Space에 있는 가능한 모든 경우를 대입해 보는 경우
- 이론상으로 대부분의 암호 알고리즘과 인증을 뚫을 수 있다.
  - 근본적으로 차단할 수 있는 방법은 거의 없다. (ex: OTP)
- 방어: 시간이 오래 걸리도록 하자!
  - 암호 길이를 길게 한다, 가능한 key space를 늘린다(특수문자 사용 등)
  - 시간 당 대입 횟수를 제한한다

Dictionary Attack

- 대부분의 사용자들은 단어, 본인 이름, 생일 등을 조합하여 암호 생성
  - ex: samjo24, kim1212, kimhandsomeguy
- 이미 존재하는 단어들과 이를 조합한 것들을 먼저 넣어보자!
- 방어: 사전에 있는 단어로 비밀번호를 만들지 않기

fail2ban

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

- **apt-get install fail2ban**
- **cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local**
- **vi /etc/fail2ban/jail.local**

- https://www.digitalocean.com/community/tutorials/how-to-install-and-use-fail2ban-on-ubuntu-14-04

Log System

- 기본적으로 /var/log/ 에 들어있음
  - 시스템 로그: syslog
  - 보안 로그: auth.log
  - 크론 로그: cron
  - 부팅 로그: boot.log
  - 웹 로그: apache2/

- Linux 배포판마다 로그 파일 이름이 조금씩 달라짐
  - ex: CentOS에서는 auth.log가 아니라 secure임

File Permission – setuid, setgid

- 400? 700? 644? 777?
  - 기본적인 것들은 다 아실 거라고 믿습니다.

- setuid / setgid: Set User / Group ID upon execution
  - /etc/passwd 파일은 해시된 비밀번호를 저장하고 있으며 root에게만 쓰기 권한이 있다. 하지만 모든 사용자는 passwd 프로그램을 통해 본인의 비밀번호를 바꿀 수 있다...???
  - setuid가 설정되어 있는 파일은 실행시 소유자의 권한으로 실행된다!
  - setgid가 설정되어 있는 파일은 실행시 소유 그룹의 권한으로 실행됨

  - chmod 4xxx – setuid, -rw**s**r--r--
  - chmod 2xxx – setgid, -rwxrw**s**r--

File Permission – sticky bit

- /tmp, /var/tmp: permission 777
  - 모든 사람이 파일을 만들고 수정하고 삭제할 수 있다
  - 사용중인 파일을 지워버리면 서비스에 장애가 발생할 수 있음

- Sticky Bit: 이 bit가 설정된 directory 안에서는 파일의 소유자만이 삭제 가능 & directory의 소유자만이 삭제 가능

- chmod 1xxx – sticky bit, drwxrwxrwx**t**

Firewall (iptables)

- 우리가 모두 알고 있는 방화벽이다.
  - 특정 조건에 맞는 패킷들을 허용하거나 거부한다.
- 3가지의 미리 정의된 chain이 존재하여 규칙들을 수행한다
  - INPUT: 들어오는 패킷 / OUTPUT: 나가는 패킷 / FORWARD: 통과하는 패킷
- 여러 가지 조건들을 사용하여 패킷을 필터링 할 수 있다.
  - source, destination, protocol, state, ...
- 매치된 패킷에 대해 다양한 동작을 취할 수 있다.
  - ACCPET, DROP(버린다), REJECT(거부되었다는걸 알려준다), ...

Firewall (iptables)

- 네트워크 연결 상태에 따라 필터링할 수도 있다.
  - NEW: 새로운 연결을 요청하는 패킷(ex: http)
  - ESTABLISHED: 기존 연결의 일부인 패킷
  - RELATED: 기존 연결에 속하지만 새로운 연결을 요청하는 패킷
    - ex: 접속포트가 20인 FTP가 1024를 이용하여 전송하고 싶어한다.
  - INVALID: 나머지

- 명령어
  - -A: 규칙 추가, -D: 규칙 삭제, -L: 규칙 출력, -P: 기본 정책 변경 …

Examples of iptable rule

- 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 -i eth0 -p tcp --dport 25 -m state --state NEW,ESTABLISHED -j ACCEPT
- iptables -A OUTPUT -o eth0 -p tcp --sport 25 -m state --state ESTABLISHED -j ACCEPT

- 명령어가 아주 복잡하고 길다. 타이핑하기 귀찮다.
- 좀 더 빠르고 쉽게 방화벽을 설정해보자 => “ufw”

ufw (Uncomplicated FireWall)

- 설치: **apt-get install ufw**
- 켜고 끄기: **ufw [enable | disable]**
  - **지금 실행하지 하지 마세요**
- 상태 보기: **ufw status**
- Rule 추가하기: **ufw [allow | deny | reject] <rule syntax>**
  - <rule syntax> ::= <port_no>, <port_no>/<protocol>, <service_name> from <ip_range> to <ip_range>
  - ex: 53 (tcp 53과 udp 53이 모두 포함), 25/tcp, http (tcp/80)
  - ex: from any to 127.0.0.1, from 192.168.0.0/16 to any
  - c.f: 0.0.0.0 – any, 127.0.0.1 – localhost
- Rule 삭제하기: **ufw delete [allow | deny | reject] <rule syntax>**

Tutorial

- Tutorial 목표: ssh와 http를 허용하는 룰을 추가하고, 방화벽을 켜자!

- Tutorial 과정:
  - 1. **ufw allow ssh**
  - 2. **ufw allow http**
  - 3. **ufw enable**
  - 4. **ufw status**

KAIST Firewall

- KAIST IP Range: 143.248.0.0/16

1. 원내 접근 차단 정책 (2013.3.15 현재)
   - Inbound 차단 정책
   o 서버관련 포트 : FTP(21), SSH(22), Telnet(23), Xwindows(6000), SNMP(161), MS-SQL(1433)
   o 원격데스크탑 : RDP(3389), VNC(5900)
   o 네트워크관련 포트 : ICMP-all
   o 유해 포트 :
   TCP-42,137,135,139,6969,1025,1080,1433,2745,3127,3410,4899,5000,5554,4444,6129,
   6667,6668,9898,9996,12345,12346,44444,47268,54321
   UDP-19,137,139,445,5000,6112,1080,1025,6129,6969,6667,666831337
   (유해포트의 경우, 공지없이 차단될 수 있습니다.)

- 정보통신팀에 접근 허가 신청서를 내면 사용 가능
- “거울”의 경우 KAIST 네트워크 밖에 있으므로 위 정책을 적용받지 않음

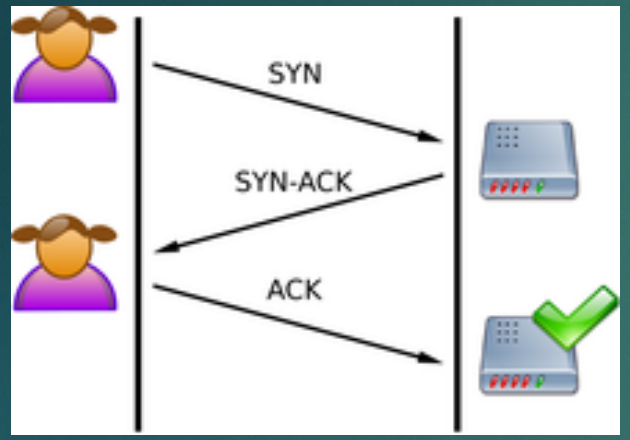
DoS / DDoS Attack

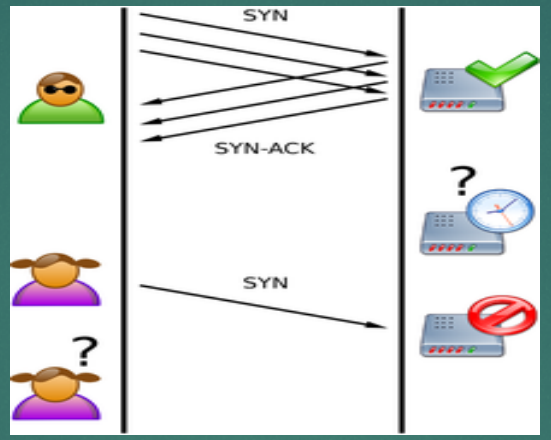
- Denial of Service / Distributed Denial of Service
- 한정된 네트워크 자원을 모두 소모시켜 정상적인 사용자가 서비스에 접근하지 못하도록 하는 공격

- 네트워크 자원을 소모시키기만 하면 되므로, 다양한 공격 방법이 존재
  - SYN Flood Attack
  - Teardrop Attack
  - Peer-to-Peer Attack
  - Distributed Attack
  - Reflected Attack
  - ...

SYN Flooding

- DoS의 가장 대표적인 공격방법이라고 할 수 있다.





- TCP 연결을 맺을 때, server는 SYN-ACK 패킷을 보내고 client에게 ACK 패킷이 오기를 기다린다.
- Client가 수많은 SYN 패킷만 보내고 ACK 패킷을 보내지 않으면, 서버는 연결을 계속 기다리게 되고 다른 SYN 패킷을 accept하지 못할 수 있다.

Web based Attack

- 웹 페이지의 입력 공간에 공격 문자열을 삽입
- GET / POST 등을 사용하여 통신되는 데이터를 변조

- ex: SQL Injection, XSS, CSRF

- **“절대, 절대 사용자의 입력을 신뢰해서는 안된다.”**
- **“DBMS의 오류 정보는 절대 사용자에게 보여져서는 안된다.”**

SQL Injection

- 서버에서 본인이 원하는 SQL 구문을 실행하는 공격 방법

- ex: SELECT uid FROM user WHERE id='{id}' AND pw='{pw}'
  - {id} = samjo, {pw} = ' OR 1=1;--

- 방어: 기본적으로 입력을 필터링해야 함
  - Django의 경우 대부분 raw SQL을 프로그래머가 작성하지 않으므로 안전
  - PHP의 경우 각별한 주의 필요! – Prepared Statements 사용 권장

XSS (Cross Site Scripting)

- 사용자가 특정 페이지를 신뢰하여 이에 접속하면 공격자가 원하는 script를 사용자가 실행하도록 하는 공격 방법

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

- 방어: 그런거 못 쓰도록 필터링하면 된다.
  - ex: <는 <로, >는 >로 escape한다.
  - ~~JavaScript를 없앤다~~

CSRF (Cross Site Request Forgery)

- 해당 웹 사이트가 사용자의 웹 브라우저를 신뢰하여, 사용자가 요청하지 않은 명령을 공격자가 보내도록 하는 공격 방법

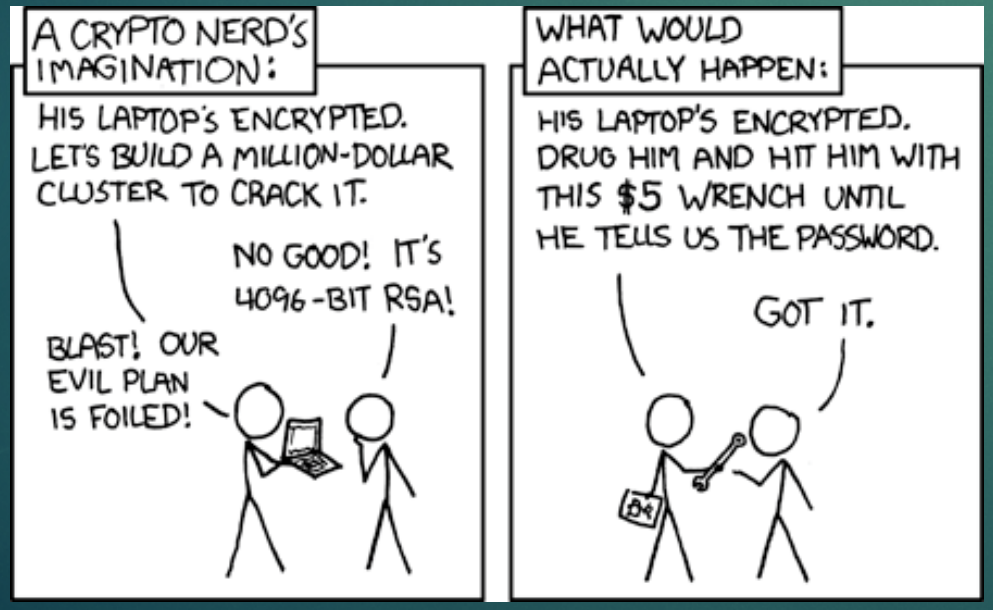
- ex: http://bank.com/withdraw?amount=백만달러&to=samjo
- 만일 이 페이지를 열어보는 사람이 bank.com에 로그인되어 있다면, 백만달러를 samjo에 보내게 될 것이다!

- 방어: 서버 쪽에서 랜덤한 token이 포함된 페이지를 사용자에게 보내고, 사용자에게서 요청을 받을 때는 token이 올바른지 확인한다.
  - Django의 csrfmiddlewaretoken

Physical Security

- 기기에 대한 물리적 접근 제한 / 통제
  - ex: Win + L, 서버실 잠금, 중요한 문서 파쇄

- 잘못되면 어떠한 암호 시스템이라도 무용지물이 될 수 있음



Social Engineering

- 각종 심리적인 방법을 사용해서 공격하는 기법
  - Phishing: “제가 사이트 관리자인데요, 님 계정에 심각한 문제가 발생했으니 비밀번호 좀 주세요, 계좌 비밀번호도 좀 주세요 다 주세요 ㅅㄱ”
  - Biting: 악성 파일이 든 CD나 USB를 길거리에 놓아둔다. 지나가던 사람이 주워서 PC에 mount하면 원하는 파일이 실행되고... PC가 망하고 사회가 망하고...
  - Tailgating: 직원 옷을 입고 너무나 자연스럽게 보안 구역을 돌아다닌다.

CERT

- Computer Emergency Response Team (침해사고 대응팀)
- 주요 IT 관련 기관 / 회사마다 존재하며 침해사고를 담당
- KAIST CERT는 창의관 옆 본관에 존재함
- 침해사고가 발생하면 정보통신팀에 연락하면 됨

Q&A

END OF THIS SEMINAR

References

- http://en.wikipedia.org/
- https://xkcd.com/538/
- http://sparcs.org/seminar/
  - 2014 Wheel Seminar – Security by protos
  - 2013 Wheel Seminar – 12. Security by apple
  - 2012 Wheel Seminar – 18. Security by undead
  - 2011 Wheel Seminar – 19. 기본 보안 by sunguard
- 2015 CS448 정보보호개론 Lecture Note