OpenSSL을 이용한 SEED 활용 매뉴얼 v1.0

2007. 11.

안전한 IT 세상을 만드는

한국정보보호진흥원

- 차 례 -

1. 개요

국제적인 오픈 소스 커뮤니티인 OpenSSL Project가 제공하는 툴킷 OpenSSL 0.9.8f 버전에 SEED가 탑재되었습니다. 이로 인해 SEED를 이용한 제품의 개발이 용이해지고, OpenSSL을 사용하는 Opera, Safari와 같은 웹브라우저, Apache 웹서버 등 다양한 환경에서 SEED를 사용할 수 있게 됩니다.

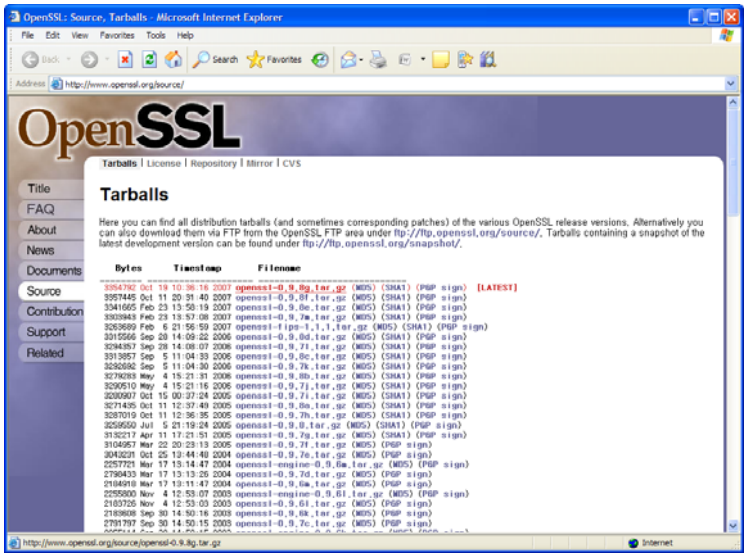
현재 최신 버전의 OpenSSL 0.9.8g에는 SEED가 탑재되어 있으나, 기본적으로는 사용 불가능한 상태입니다. 본 매뉴얼은 OpenSSL 0.9.8g에서 SEED를 활성화하고, 활성화된 SEED를 이용하여 암호화 및 복호화 하는 것을 도와드리는 설치 및 활용 가이드입니다. 여기서는 윈도우 환경에서 SEED의 작동을 확인할 수 있도록, Cygwin[1] 상에서의 설치 방법을 예로 들어 설명하고 있습니다만, Linux나 Solaris 등의 환경에서도 기본적으로는 같은 방법으로 설치할 수 있습니다.

2. 설치

현재 최신 버전의 OpenSSL 0.9.8g에는 SEED가 포함되어 있으나, 기본적으로는 사용이 불가능한 상태이기 때문에, OpenSSL 0.9.8g 설치 시 SEED를 활성화한 후 설치하셔야만 SEED를 사용할 수 있습니다.

가. OpenSSL 다운로드

OpenSSL의 홈페이지(http://www.openssl.org/source)로부터 최신 버전 OpenSSL 0.9.8g를 다운로드합니다.



1) Cygwin: Windows 환경에서 GNU 툴을 설치하여 Linux와 같은 Look-and-Feel을 지원하는 툴 모음. Linux와 라이브러리 수준에서 매우 유사하므로, Linux 기반 프로그램을 포팅하여 실험할 때 유용한 대안입니다. 보다 자세한 내용은 Cygwin 프로젝트 사이트(http://cygwin.com)을 참고 하십시오.

나. 아카이브(Archive) 전개

다음과 같이 다운로드한 OpenSSL을 전개합니다.

```
$ gzip -d openssl-0.9.8g.tar.gz
$ tar xvf openssl-0.9.8g.tar
$ cd openssl-0.9.8g
```

다. SEED 활성화

현재 OpenSSL의 최신버전 0.9.8g에는 SEED가 추가되어 있으나, 활성화되어 있지 않으므로 OpenSSL 설치 전에 다음과 같이 SEED를 활성화시켜 주어야 합니다.

```
$ ./config enable-seed
   [shared library의 경우에는 다음과 같이 옵션을 부여합니다.
    $ ./config enable-seed shared]
$ make depend
```

라. 컴파일 및 설치

다음과 같이 컴파일 및 설치를 진행합니다. 이 때, 명령어, 파라미터 등은 환경에 따라 다를 수 있으므로, 사용하고 있는 환경에 맞춰 주시기 바랍니다.

```
$ make
$ make test        <- 컴파일이 제대로 되었는지 확인하는 절차이므로, 생략해도 됨
$ make install
```

마. 결과 확인

설치가 정상적으로 종료되면, 다음과 같이 SEED가 사용 가능한 지 확인합니다. Cygwin의 경우 openssl.exe가 usr/local/ssl/bin 하에 작성됩니다. 아래의 예에서는 OpenSSL에서 사용 가능한 암호수트(CipherSuite)[2]를 나타내고 있습니다.

```
$ cd /usr/local/ssl/bin
$ openssl ciphers
…(생략)…:DHE-RSA-SEED-SHA:DHE-DSS-SEED-SHA:SEED-SHA:…(생략)…
```

2) 암호수트(CipherSuite): SSL 연결이 인증, 키 교환, 암호화 통신 등을 실시하기 위해 사용하는 암호 알고리즘의 조합

3. OpenSSL을 통한 SEED 활용

OpenSSL에서는 명령어 및 API의 형태로 암・복호화를 수행할 수 있습니다.

가. 명령어를 이용하는 방법

대칭키 암호를 이용하여 문서를 암호화할 때, 주로 enc 옵션을 이용합니다. 사용하는 알고리즘은 암호알고리즘, 키길이 및 운영모드를 조합하여 사용하는데, SEED의 경우 키길이가 128비트 이므로 이를 생략하고 다음의 옵션을 사용합니다.

seed-ecb seed-cbc seed-cfb seed-ofb

다음은 명령어를 이용하여 SEED를 활용하는 예제입니다.

1) enc 명령어 이용

```
$ ./openssl enc -seed-cbc -in test.txt -out test_out.bin
enter seed-cbc encryption password:
Verifying - enter seed-cbc encryption password:
```

위와 같이 test.txt를 SEED의 CBC모드로 암호화하여 test_out.bin의 암호문을 생성하면 패스워드를 지정하라는 메시지가 뜨게되고, 확인 절차를 거친 후 암호문 test_out.bin이 생성됩니다. 생성된 test_out.bin 파일은 아래와 같이 복호화합니다.

```
$ ./openssl enc -d -seed-cbc -in test_out.bin -out new_test.txt
enter seed-cbc decryption password:
```

2) 암호 방식명을 옵션으로 이용

```
$ ./openssl seed-cbc -in test.txt -out test_out.bin
enter seed-cbc encryption password:
Verifying - enter seed-cbc encryption password:
$
$ ./openssl seed-cbc -d -in test_out.bin -out new_test.txt
enter seed-cbc decryption password:
```

openssl 명령의 보다 자세한 설명은 OpenSSL 관련 자료를 참조하시기 바랍니다.

나. API를 이용하는 방법

OpenSSL에서는 EVP API를 이용하여 암호 알고리즘이나 방식 등에 상관없이 공통의 인터페이스에서 프로그래밍을 할 수 있습니다.

본 절에서는 [첨부 2]의 예제 코드를 통해 EVP API를 이용하여 SEED를 활용하는 방법을 살펴봅니다. 보다 자세한 내용은 OpenSSL/doc 디렉토리에 있는 파일들을 참고하시기 바랍니다.

1) 암호 콘텍스트(context)[3]의 정의와 초기화

먼저 암·복호화에 필요한 암호 콘텍스트를 **EVP_CIPHER_CTX_init** 함수를 이용하여 초기화 합니다.

```
EVP_CIPHER_CTX enc, dec;        // 암호 콘텍스트의 정의
                                   enc : 암호화용
            ⋮                      dec : 복호화용

EVP_CIPHER_CTX_init(&ctx);      // 암호 콘텍스트 초기화
```

2) 암·복호화 실행을 위한 초기 설정

암·복호화를 실행하기 위한 정보를 **EVP_CipherInit** 함수를 이용하여 암호 콘텍스트에 설정합니다.

변 수	설 명
cipher	사용하는 암호의 종류
key	암·복호화에 사용하는 키
iv	초기치 벡터
kind	암호화 / 복호화

```
ret = EVP_CipherInit(&ctx, cipher, key, iv, kind);
if(ret == 1) {
            ⋮                       // 처리 결과 판정
}
```

[첨부2]의 예제 코드에서는 각각의 파라미터를 개별적으로 설정하기 때문에 여러 번 **EVP_CipherInit** 함수를 호출하지만, 한 번에 설정하는 것도 가능합니다.

---

3) 암호 콘텍스트: 데이터를 암·복호화 하기 위해 필요한 정보를 저장하는 데이터 구조체. EVP_CIPHER_CTX_init 함수를 이용하여 초기화하고, EVP_CIPHER_CTX_cleanup 함수를 이용하여 사용한 구조체를 환원

3) 암・복호화 실행

다음으로, **EVP_CipherUpdate** 함수를 이용하여 평문을 암호화합니다. 매개변수는 다음과 같습니다.

변 수	설 명
enc_data	암호화된 데이터가 저장될 변수
enc_len	암호문의 길이
plain_data	패딩된 평문 데이터
sizeof(plain_data)	평문의 길이

```
/* 평문(plain_data)을 암호화한다
ret = EVP_CipherUpdate(&ctx, enc_data, &enc_len,
      plain_data, sizeof(plain_data));
if(ret != 1)  {
      :
}
/* 암호화 최종 블록을 처리한다
ret = EVP_CipherFinal(&ctx, enc_dara + enc_len,
      &tmplen);
if(ret != 1)  {
            ⋮                          // 최종 블록에 대한 처리
}
```

최종 블록에서, 블록 크기를 채우지 못한 데이터는 다음 **EVP_CipherUpdate** 함수가 호출되거나, **EVP_CipherFinal** 함수가 호출되지 않는 한, 암호화가 더 이상 진행되지 않습니다.

4) 컴파일 및 실행

작성한 소스코드(test.c)를 실행하기 위해서는 **gcc**와 같은 컴파일러를 이용하여 실행파일을 생성해야 하는 데, 이때 **crypto** 라이브러리를 설정해 주어야 합니다.

```
$ gcc -o test test.c -lcrypto
$ ./test
            ⋮
```

[첨부1] 함수 목록

[첨부2] 예제코드에서 사용하고 있는 주요 함수들의 목록과 간단한 설명을 아래와 같이 제시합니다.

No.	함수명	기능
1	EVP_CipherInit	대칭키 알고리즘을 사용하여 암・복호화를 실행하기 위해, 알고리즘, 키, IV(초기화 벡터) 등을 암호 콘텍스트에 설정
2	EVP_CipherUpdate	대칭키 알고리즘을 사용하여 데이터의 암・복호화를 실시
3	EVP_CipherFinal	암・복호화를 실시한 결과 데이터의 최종 블록을 읽음
4	EVP_CIPHER_CTX_init	암호 콘텍스트를 초기화
5	EVP_CIPHER_CTX_cleanup	암호 콘텍스트를 지움
6	EVP_CIPHER_CTX_block_size	암호문을 저장하는 블록의 크기(바이트 길이)를 얻음
7	EVP_CIPHER_CTX_iv_length	IV의 크기(바이트 길이)를 얻음
8	EVP_CIPHER_CTX_set_padding	암호 콘텍스트에 패딩 유무를 설정
9	OpenSSL_add_all_algorithms	전체 알고리즘의 EVP 오브젝트와 문자열을 관련지은 내부 테이블을 읽어 들임
10	CRYPTO_cleanup_ex_data	crypto 라이브러리의 내부 영역을 해방
11	RAND_poll	PRNG(Pseudo Random Number Generator, 의사 난수 생성기)에 시드[*1]를 추가
12	RAND_seed	작동 환경 등에 의해 PRNG에 충분한 시드가 부여되지 않을 때, 본 함수로 PRNG에 시드를 추가
13	RAND_cleanup	PRNG를 지움
14	RAND_bytes	의사 난수를 생성
15	ERR_load_crypto_strings	암호 모듈용 에러 메시지를 읽어 들임
16	ERR_free_strings	읽어 들인 에러 메시지 영역을 해방
17	ERR_remove_state	쓰레드가 가지는 에러 큐를 해방

*1 시드란 PRNG의 초기 상태 설정에 사용되는 예측하기 어려운 비밀 데이터 값

[첨부2] 예제 코드

```
/* SEED EVP API를 이용한 대칭키 암호화·복호화의 샘플프로그램 */

#include <stdio.h>
#include <string.h>
#include <openssl/evp.h>
#include <openssl/err.h>
#include <openssl/rand.h>
#include <openssl/crypto.h>

#if defined(_MSC_VER) && defined(_WIN32)
#endif
#define PLAIN_DATA_SIZE 32
#define NO_PAD 0
#define PAD 1
#define UNDEF -1
#define NO_IV 0
#define SET_IV 1
#define ENCRYPT 1
#define DECRYPT 0

/*
* --<binary data 표시>--
* 암호문 등의 바이너리 데이터를 표시
*/
void binary_print_data(const unsigned char* data, int len, char* msg)
{
        Int i;
        printf("%s:\n",msg);
        for (i=0; i<len; i++) {
                printf("0x%02x, ", data[i]);
                if (((i+1)&0x7) == 0) {
                        printf("\n");
                }
        }
        printf("\n");
}
```

```
/*
*--<암호화·복호화처리>--
* 파라미터로 지정된 암호알고리즘을 이용하여
* 평문을 일괄적으로 암호화하고 암호문을 일괄적으로 복호화함
*
* <인수(argument)>
* alg : 대칭키 암호알고리즘
* keylen : 키길이(byte 사이즈)
* setiv : IV(IV 설정 필요함), NO_IV(IV 설정 필요 없음)
* pad: PAD(패딩 있음), NO_PAD(패딩 없음)
*
*<반환값>
*1: 정상 종료
*0: 비정상 종료
*/

int sample_scenario(const EVP_CIPHER *cipher, int keylen, int setiv, int pad)
{
        int ret = 0;
        EVP_CIPHER_CTX enc,dec;
        unsigned char *key = NULL;
        unsigned char *iv = NULL;
        int ivlen = 0;
        const unsigned char plain_data[PLAIN_DATA_SIZE] = {
                                0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,
                                0x08,0x09,0x0a,0x0b,0x0c,0x0d,0x0e,0x0f,
                                0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77,
                                0x88,0x99,0xaa,0xbb,0xcc,0xdd,0xee,0xff};
        unsigned char *enc_data = NULL;
        unsigned char *dec_data = NULL;
        int enclen;
        int declen;
        int tmplen;

/************ 여기부터 암호화처리**************/
        EVP_CIPHER_CTX_init(&enc);
        ret = EVP_CipherInit(&enc, cipher, NULL, NULL, ENCRYPT);
```

```
/* 암호화에 사용하는 EVP_CIPHER_CTX를 초기화 */
        if(ret != 1) {
                printf("EVP_CipherInit failed!\n");
                goto cleanup;
        }

/* 주키의 생성(여기에서는 난수를 이용) */
        key = (unsigned char*)OPENSSL_malloc(keylen);
        RAND_bytes(key,keylen);
        binary_print_data(key,keylen,"key");

/*
* IV 생성(여기에서는 난수를 이용)
* ECB 모드에는 IV를 사용하지 않기 때문에 설정할 필요가 없음
*/
        if(setiv == SET_IV)
        {
                ivlen = EVP_CIPHER_CTX_iv_length(&enc);
                iv = (unsigned char*)OPENSSL_malloc(ivlen);
                RAND_bytes(iv,ivlen);
                binary_print_data(iv,ivlen,"iv");
        }

/* 주 키·IV를 EVP_CIPHER_CTX로 설정 */
        ret = EVP_CipherInit(&enc,NULL, key, iv, ENCRYPT);
        if(ret != 1) {
                printf("EVP_CipherInit failed!\n");
        }

/*
* 패딩설정
* OFB, CFB 모드는 패딩과 관계가 없기 때문에 설정 자체가 필요 없다
*/
        if(pad != UNDEF)
        {
                ret = EVP_CIPHER_CTX_set_padding(&enc, pad);
                if(ret != 1) {
                        printf("EVP_CIPHER_CTX_set_paddingfailed!\n");
```

```
                        goto cleanup;
                }
        }

/* 암호화 결과 출력 버퍼 초기화 */
        if(pad == PAD) {
                printf("padding:ON\n");
                tmplen = EVP_CIPHER_CTX_block_size(&enc);
                tmplen = ((PLAIN_DATA_SIZE + tmplen)/tmplen) * tmplen;
        }else{
                printf("padding:OFF\n");
                tmplen = PLAIN_DATA_SIZE;
        }
        enc_data = (unsigned char*)OPENSSL_malloc(tmplen);
        memset(enc_data, 0, tmplen);
        tmplen = 0;

/* 암호화 */
        ret = EVP_CipherUpdate(&enc, enc_data, &enclen, plain_data, sizeof(plain_data));
        if(ret != 1) {
                printf("EVP_CipherUpdate failed!\n");
                goto cleanup;
        }

/* 암호화 최종 블록 처리 */
        ret = EVP_CipherFinal(&enc, enc_data + enclen, &tmplen);
        if(ret != 1) {
                printf("EVP_CipherFinal failed!\n");
                goto cleanup;
        }
        enclen += tmplen;
        binary_print_data(plain_data, PLAIN_DATA_SIZE, "plain_data");
        binary_print_data(enc_data, enclen, "enc_data");

/************ 여기부터 복호화 처리**************/
/* 복호화에 사용하는 EVP_CIPHER_CTX 구조체 초기화 */
        EVP_CIPHER_CTX_init(&dec);
```

```
/* EVP_CIPHE_CTX 구조체 설정 */
        ret = EVP_CipherInit(&dec,cipher,key,iv, DECRYPT);
        if(ret != 1) {
                printf("EVP_CipherInit failed!\n");
                goto cleanup;
        }

/* 패딩설정 */
        ret = EVP_CIPHER_CTX_set_padding(&dec, pad);
        if(ret != 1) {
                printf("EVP_CIPHER_CTX_set_padding failed!\n");
                goto cleanup;
        }

/* 복호화 결과 출력 버퍼 초기화 */
        dec_data = OPENSSL_malloc(enclen);
        memset(dec_data, 0, enclen);

/* 복호화 처리 */
        ret = EVP_CipherUpdate(&dec, dec_data, &declen, enc_data, enclen);
        if(ret != 1) {
                printf("EVP_CipherUpdate failed!\n");
                goto cleanup;
        }
/* 최종 블록 처리 */
        ret = EVP_CipherFinal(&dec, dec_data + declen, &tmplen);
        if(ret != 1) {
                printf("EVP_CipherFinal failed!\n");
                goto cleanup;
        }
        declen += tmplen;
        binary_print_data(dec_data, declen, "dec_data");

/* 복호화 결과 점검 */
        if(!(declen == PLAIN_DATA_SIZE && memcmp(plain_data,dec_data,declen) == 0)) {
                printf("Decryption is failed!\n");
                ret = 0;
                goto cleanup;
```

```
        }
        cleanup:

/* EVP_CIPHER_CTX 클린업 */
        EVP_CIPHER_CTX_cleanup(&enc);
        EVP_CIPHER_CTX_cleanup(&dec);

/* 암호화 복호화 결과 출력 버퍼 해방 */
        if(key)OPENSSL_free(key);
        if(iv)OPENSSL_free(iv);
        if(enc_data)OPENSSL_free(enc_data);
        if(dec_data)OPENSSL_free(dec_data);

/* 에러 정보 클리어 */
        ERR_clear_error();
        return ret;
}



/*
* --<샘플 프로그램>--
* EVP_Cipher 계열 API를 이용하여 각 암호알고리즘의 작동을 확인
*
* <인수>
* alg : 대칭키 암호알고리즘
* keylen : 키길이의 크기
* setiv : IV(IV 설정 필요함), NO_IV(IV 설정 필요 없음)
* pad : PAD(패딩 있음), NO_PAD(패딩 없음)
*
* <반환값>
* 1: 정상 종료
* 0: 비정상 종료
*
*/
int main()
{

/* 알고리즘 테이블 불러 옴 */
```

```
        OpenSSL_add_all_algorithms();

/* 에러 메시지 불러 옴 */
        ERR_load_crypto_strings();

/* 의사 난수 생성기 초기화 */
        RAND_poll();
        while (RAND_status() == 0) {
                int rnd = rand();
                RAND_seed(&rnd, sizeof(rnd));
        }

/******* 암호화·복호화 처리*******/
/* SEED(CBC 모드, 패딩 없음) */
        printf("------ SEED(mod:CBC, no-padding) ------\n");
        sample_scenario(EVP_seed_cbc(),16, SET_IV, NO_PAD);

/* SEED(CBC 모드, 패딩 있음) */
        printf("------ SEED(mod:CBC, padding) ------\n");
        sample_scenario(EVP_seed_cbc(), 16, SET_IV, PAD);

/* 난수 해방 */
        RAND_cleanup();

/* 에러 메시지 해방 */
        ERR_remove_state(0);
        ERR_free_strings();

/* 전체알고리즘의 EVP 구조체와 문자열을 관련지은 데이터 삭제 */
        EVP_cleanup();

/* crypto 라이브러리의 내부 영역 해방 */
        CRYPTO_cleanup_all_ex_data();

        return 0;
}
```