정보보호개론

-Assignment 2-

2017310100 김수종

1. 개발 환경

해당 과제는 리눅스 기반인 wsl2에서 진행하였다. AES와 DES 암복호화를 위해 <openssl/des.h>, <openssl/aes.h> 라이브러리를 사용하였고 base64 encode를 위해 <openssl/bio.h>, <openssl/evp.h>를 사용하였다. 컴파일을 위해 -crypto 옵션을 사용하여

* gcc src.c -lcrypto

명령어를 사용하였다. 해당 소스코드를 컴파일 하기 위해서 소스코드와 동일한 디렉토리에 key search에 사용될 PlaintextCiphertext.txt 파일이 존재하여야 하며 또한 password 정보를 담고 있는 password.txt 파일도 존재하여야 한다. PlaintextCiphertext.txt 파일에서 Plaintext에는 줄 바꿈(‘\n’)이 없는 것으로 간주하고 첫번째 줄에 Plaintext, 그 이후 줄에 Ciphertext가 존재한다고 가정하고 과제를 진행하였다.

1. 소스 코드 작동방식

해당 과제는 plaintext를 DES로 한번 encrypt하고 AES-128로 다시 encrypt 된 결과로부터 encrypt에 사용된 두 개의 key를 찾아내는 과제이다. 해당 과제를 수행하기위해 주어진 key list는 총 184389개로 두 개의 key를 완전탐색으로 찾게 되면 184389^2의 연산이 필요하게 된다. 이는 시간 소요가 너무 크므로 수업시간에 2DES 공격방식에서 사용된 Meet in the middle 방식을 선택하였다. 이 방식은 Plaintext로부터 DES로 encrypt를 진행하고 Ciphertext로부터 AES로 decrypt를 진행한 후 두 값을 비교하는 방식으로 진행된다.

가장 먼저 소스코드는 PlaintextCiphertext.txt를 읽어서 첫째줄을 Plaintext로 그 이후를 모두 Ciphertext로 저장한다. 그 이후 Ciphertext의 base64 decode를 실행한다. 해당 과정에서 openssl을 사용하였다.

그 후 먼저 ciphertext에 대한 decrypt을 먼저 실시하였다. password.txt의 처음부터 모든 password에 대해 openssl을 통해 AES\_cbc\_128 decrypt를 실행하였다. 여기서 나온 결과값 중 가장 앞의 6바이트 만을 DP\_struct의 value 값에 long long 값으로 변경하여 저장하였다. 이는 6바이트만 저장하여도 128^6 = 4,398,046,511,104 으로 약 19만개의 password를 구분할 때 중복된 값이 나올 확률이 현저히 낮아지고 이후에 정렬 및 비교 연산에서 속도가 빨라지기에 이렇게 구현하였다. 또한 DP\_struct의 index 값에는 해당 key의 순서를 기억하여 이후에 정렬된 이후에도 기존의 key를 알 수 있게 하였다. 모든 password에 대해 AES decrypt을 끝낸 후 DP\_struct의 value 값을 기준으로 오름차순으로 정렬을 진행하였다. 해당 정렬에는 c언어 내장함수인 qsort를 이용하였다.

이렇게 모든 후보 key에 대해 AES decrypt를 진행한 후 plaintext들을 모든 key에 대해 DES로 encrypt도 진행하였다. 이번에도 마찬가지로 openssl을 통하여 des\_ecb\_encrypt를 진행하였다. 이렇게 나온 결과값 중 가장 앞의 6바이트 값을 long long 형식으로 추출해 낸 후 이 값을 기존의 저장되어 있던 DP\_struct의 value 값과 비교 연산을 하여 일치하는 값이 있는지 탐색하였다. 여기서 연산 속도 향상을 위해 이진탐색을 사용하였다. 만약 일치하는 값이 있다면 해당 값의 index 값을 확인하고 해당 index에 해당하는 password를 keys.txt 파일에 저장하고 연산을 종료하였다.

1. 분석

기존의 key 탐색 방식대로 모든 key에 대해 plaintext를 DES encrypt를 진행하고 AES encrypt를 진행한 후 ciphertext와 비교했다면 후보 key가 n 개 일 때 O(n^2)의 시간 복잡도가 소요된다. 하지만 해당 소스코드에서는 AES encrypt와 DES decrypt를 따로 진행하고 중간에서 해당 값들을 이진 탐색을 통해 비교하여 시간 복잡도는 이진탐색의 시간 복잡도인 O(nlogn)만큼 소요되었다. 실제로 해당 코드를 통하여 여러 테스트케이스를 돌려보았다.

Input: 예시 PlaintextCiphertext.txt, Time: 0.306 sec 결과 일치

Input: 임의의 50KB plaint text, Time: 33.425 sec 결과 일치

Input: 임의의 496KB plain text, Time: 210.0173562 결과 일치

예상보다 실행시간이 오래 걸려 어느 부분에서 실행시간이 긴 지 확인하고 해당 부분에 최적화를 진행하려 하였다. 하지만 대부분의 실행시간은 DES encrypt에서 발생하고 있었다. Plaintext가 길어질 때 parallel하게 연산을 진행하지 못하는 DES에서 해당 plaintext를 암호화하면서 대부분의 시간이 소요되었고 그 외의 이진탐색과 AES encrypt의 경우 빠른 시간안에 실행되었다. 또한 DES encrypt를 최상단의 password부터 비교하며 일치하는 값이 있는 경우에는 결과값을 출력하고 종료되므로 앞쪽에 있는 key를 DES encrypt에 사용할수록 실행시간은 짧아졌다.