[2023 JBUCTF] crypto

ecb\_mode

Write-Up

문제 개요

제공 파일 : ecb\_mode.py

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

Base64 인코딩 된 문자열을 입력으로 받고, 그 문자열을 base64 디코딩하여 **flag와 더한 후**, 패딩을 하고, 동일한 key를 사용해서 AES-ECB로 암호화 한 결과를 다시 base64 인코딩해서 출력해준다.

그리고 이를 반복한다.

패딩

# Wikipedia Padding (cryptography) : <https://url.kr/v76dwc>

# IBM PKCS padding method : <https://url.kr/rblvmy>

pad 함수의 동작은 입력 받은 문자열이 n개의 바이트로 이루어져 있을 때, 바이트의 개수가 입력

받은 블록의 크기의 배수가 되도록 특정한 규칙을 가진 값을 채운다.

( 문제에서는 블록의 크기가 16이므로 16을 기준으로 설명 하겠다. )

n을 16으로 나눈 나머지의 값을 x라고 하면 (16 – x)를 x번 만큼 뒤에서 채운다.

만약 n이 16의 배수이면 16을 16번 채운다.

예를 들어 010203040506 을 패딩 처리한다고 했을 때,

010203040506 의 바이트 개수가 6개 이므로 16 – 6 = 10 을 10번 채운다.

즉 010203040506 을 패딩 처리하면

0102030405060a0a0a0a0a0a0a0a0a0a 가 된다.

AES-ECB 모드

Wikipedia 전자 코드북 (ECB) : <https://url.kr/tpevx7>

AES-ECB 운영모드는 각각의 16byte 블록마다 암호화를 적용하는 모드이다.

암호문 블록들 서로의 연관성이 없으므로 동일한 16byte 평문을 암호화 한 암호문 블록들의 결과값은 서로 같다.

이 문제에서는 이러한 원리를 이용하여 암호화된 flag의 값을 알아낸다.

문제 풀이

flag의 길이는 평문이 16byte 단위로 패딩 되는 것을 이용해서 구한다.

입력 값의 크기를 1 ~16 만큼 줘서 입력하다 보면 암호문의 크기가 변하는 순간이 있다. 그 때의 입력 값의 크기를 변하지 않았던 암호문의 길이에서 빼면 flag의 길이가 된다.

1 ~ 16 길이만큼의 입력 값을 넣었는데도 암호문의 크기가 변하지 않았다면 암호문의 길이에

1 + 16을 뺀 값이 flag의 길이이다.

실제로 적용해보면 flag의 크기가 48byte임을 알 수 있다.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

아래의 이미지는 flag가 16byte인 하나의 예시로 암호문의 평문 값을 알아내는 과정의 일부이다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 번호, 라인이(가) 표시된 사진

자동 생성된 설명

입력 값을 블록의 크기인 16byte가 아닌 17byte로 하면 암호화되는 값이 16 + 16 + 1 이 되므로

마지막 블록의 첫 1byte는 0~255인 하나의 값과 15개의 0f로 패딩 된 평문이 암호화되었다는 것을 알 수 있다. 이 블록을 B라 하자.

텍스트, 스크린샷, 라인, 직사각형이(가) 표시된 사진

자동 생성된 설명

따라서 0~255 범위의 값을 A라 할 때, 첫 바이트를 A로 두고, 나머지 값은 0f로 16byte가 되게 해서 (A + 0f\*15)를 입력 값에 입력해서 암호화 시킨다.

암호화 된 결과가 블록 B와 같은 지 확인하고, 같다면 A의 값이 flag의 마지막 byte인 것이다.

이를 반복해서 모든 flag의 값을 구하면 된다.

아래의 코드는 이를 자동화하여 구현한 것이다.

exploit.py

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

FLAG

scpCTF{42533c6a4e762e17bfe825c43331d7fb25842cb0}