Report 5

<https://bpsecblog.wordpress.com/2016/10/07/oldschool_timeattack/> 사이트를 통해 타이밍어택의 원리와 기본 코드를 참고하여 과제 수행을 진행하였다.

1. 처음에는 해당 사이트를 기반으로 socket 함수를 이용하여 docker와 통신하니 0.01 ~ 0.1 초 사이의 통신이 이루어졌다. 직접 입력할때에는 5초 이상걸리는 답변에 비해, 극단적으로 짧은 시간이었다. 이를 해결하기 위해, socket 에 대해 조금 더 구글링 해보았고, 이를 통해 socket에 대해 이해하고 해결법을 찾게 되었다.

Python의 s.recv 함수는 서버측의 \n까지의 응답을 받아오는 함수라서 시작 문자열인 Start로 인해 위와 같은 현상이 발생했다. 이는 s.recv를 이용해 Start 문자열을 한번 받아 온 뒤, send 한 뒤 다시 recv를 이용함으로써 해결하였다.

3. 맨 처음, flag 형식에 기반하여 0~999999까지 for문을 진행하였다. 하지만 직접 실행시켜보니 응답 1번당 최소 5초에서 최대 15초까지 걸리는 특성상 이는 불가능 하다고 판단하였다.

그 후, flag를 0~10까지의 for문을 이용하여 총 6번 반복하는 2중 for문을 활용하게 되었고, 확인된 값을 str형태로 바꾸어 flag 형식에 + 연산하여 값을 추출하였다.

4. Basetime(FLAG가 모두 일치하지 않을 때 응답시간)을 코드 시작 부분에 작성하여 프로그램을 시작하니, FLAG가 하나씩 맞춰질때마다 응답시간이 길어지고, Basetime이 변화하였다. 이는, Basetime 추출 부분을 코드 시작 부분이 아닌 for문에 삽입하여 해결하였다.

5. if (dtime(send후 응답시간) > basetime) 구문을 활용하여 basetime보다 시간이 많이 걸린 응답이 flag라고 판단하는 코드를 작성하였다. 하지만, 0.000…001의 오차라도 발생하면 flag로 판단하였고, 이는 잘못된 flag를 출력하는 문제를 야기하였다. 이는, 위의 구문에서 basetime에 + 0.5를 하여 해결하였다.

6. 이후 코드를 실행 시켜 HCAMP{031216}을 send하여 Submit the Flag!를 recv 함으로써 올바른 flag를 획득하였음을 확인하였다.

텍스트, 스크린샷, 폰트, 메뉴이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명