stkof write-up by jaehyeon

2019-01-18

Background

- heap 상식
- unsafe unlink

Binary Analysis

이전에 푼 Sleepy Holder 랑 구성이 비슷하다.

alloc, fill, wipe 함수들로 메모리를 할당하고, 할당된 메모리에 값을 쓰고, 할당된 메모리를 해제할 수 있다.

Sleepy Holder 랑 다르게 wipe 시 free() 호출 후 해당 변수를 0으로 초기화 하기 때문에 double free 가 쉽사리 일어나지는 않는다.

대신 fill 에서 size checking 을 하지 않으므로 heap overflow를 발생시킬 수 있다. 별도의 dump 함수가 없으므로 babyheap 문제와 같이 main_arena leaking은 힘들 것 같고 PIE, Full relro 가 아니니 unsafe unlink 를 시도 할 수 있다.

근데.....

Exploit

Sleepy Holder 에서의 exploit 방법과 너무나도 똑같은 방식으로 unsafe unlinking이 진행된다. 이걸 풀고 Sleepy Holder를 풀어야 한다. Sleepy Holder 와 달리 쓸 수 있는 전역 변수도 무제한이고 fastbin_dup_consolidate 도 필요 없다.

Heap overflow 가 발생하니까 unsafe unlink 진행하기가 너무나도 수월하다.

Sleepy Holder 에서 malloc_consolidate() 를 호출한 이유가 large_chunk 의 prev_in_use bit를 0으로 만들려고 했던 것인데 heap overflow 환경에서는 그냥 때려 넣으면 되니까 너무나도 편하다.

unlinking 세부사항을 보려면 [sleepy holder write-up] 을 보자

fake chunk의 size도 똑같고 unlinking 이후 leaking 과정도 똑같다.

Useful Information

- program 실행하고 처음 stdout 에 내용을 쓰는 함수를 호출할 때 (ex. puts, printf) 내부적으로 setvbuf() 같은 것을 호출해서 buffer 로 사용할 메모리를 힙으로부터 할당 받는 듯 하다. 첫 번째 printf 호출 이 후 heap을 봤더니 0x210 size의 chunk가 할당되어 있고 size field 이후 (user data 영역)부터 printf 의 format string이 저장되어 있는 것을 확인하였다. 첫 번째로 stdout 으로 write 하는 함수를 호출할 때 malloc(512)를 호출하는 것으로 추측된다. (buffer를 쓰는 함수의 경우에만 해당, vfprintf source 를 보면 malloc() 호출을 볼 수 있다.)
- 따라서 exploit 을 위해 alloc(0x30-0x8), alloc(0x200-0x8) 을 호출하면
- chunk1:0x30 할당 -> stdout buffer:0x210 할당 -> chunk2: 0x200 할당 이 순서로 할당되므로 chunk1, chunk2가 인접하지 않다.
- 즉 chunk1을 할당하기 전에 puts나 printf 를 먼저 호출해서 buffer 를 먼저 만들어 두고 다음에 chunk1, chunk2를 할당하자.

reference

- x

environment information

- In WSL!
- glibc version : libc-2.23.so (do not use tcache)
- used socat to run the program in the port 10003
- debugging with peda

exploit code

- link (github)