**24.01.18 보고서**

**서강대학교 20191244 김현승**

* **Exploit Dact**

Dump of assembler code for function dact\_process\_file:

0x0000555555557f80 <+0>: endbr64

0x0000555555557f84 <+4>: push %r15

0x0000555555557f86 <+6>: mov $0x800,%r10d

0x0000555555557f8c <+12>: mov %esi,%r15d

0x0000555555557f8f <+15>: push %r14

0x0000555555557f91 <+17>: mov %r8,%r14

0x0000555555557f94 <+20>: push %r13

0x0000555555557f96 <+22>: push %r12

0x0000555555557f98 <+24>: push %rbp

0x0000555555557f99 <+25>: mov %edi,%ebp

0x0000555555557f9b <+27>: push %rbx

0x0000555555557f9c <+28>: mov %edx,%ebx

0x0000555555557f9e <+30>: sub $0x918,%rsp

위 어셈블리 코드는 dact\_process\_file 함수에서 스택에 callee-saved register 값들을 저장하고 다른 지역 변수들을 위한 공간을 만드는 부분의 코드이다. 6개의 callee-saved register를 push한 후 0x918 바이트만큼 스택 공간을 할당하였다. 따라서 dact\_process\_file 스택 프레임에서 %rsp 값이 N이라고 했을 때, return address는 N+0x918+(6\*0x8) = N+0x948에 저장되어 있다. 스택 BOF를 발생시키는 file\_extd\_urls 배열이 스택의 어느 위치에 저장되는지 알아보자.

=> 0x0000555555558b89 <+3081>: mov 0x68(%rsp),%rdi

0x0000555555558b8e <+3086>: call 0x55555555ba60 <parse\_url\_subst>

0x0000555555558b93 <+3091>: mov 0x68(%rsp),%rdi

0x0000555555558b98 <+3096>: mov %rax,0x110(%rsp,%r12,8)

위 코드는 취약점이 발생하는

file\_extd\_urls[file\_extd\_urlcnt++]=parse\_url\_subst(hdr\_buf,filename);

부분의 어셈블리 코드이다. parse\_url\_subst(hdr\_buf, filename)의 리턴 값인 %rax 값이 %rsp + 8 \* %r12 + 0x110 에 저장된다. 여기서 offset 8은 char형 포인터 배열인 file\_extd\_urls의 원소 하나 크기이고, %r12는 file\_extd\_urlcnt 값이다. 따라서 file\_extd\_urls[0]은 N+0x110에 저장되어 있다. 현재 dact\_process\_file의 스택 프레임을 대략적인 그림으로 표현하면 다음과 같다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

N+0x910부터 총 0x38바이트를 채우고 다음 8바이트에 임의의 return address를 입력하면 성공적으로 control hijacking이 가능하다. 그런데 여기서 file\_extd\_urls[i] 들이 갖는 값은 parse\_url\_subst() 함수의 리턴 값[[1]](#footnote-1)인데, 이 함수는 hdr\_buf에 담긴 내용을 파싱해 적절한 치환 과정을 거친 뒤 동적 할당한 메모리에 치환한 문자열을 저장하고 그 주소를 반환하고 있다. 따라서 file\_extd\_urls[i]의 값들은 힙 영역 메모리 주소 값이다. 따라서 return address로 입력될 힙 영역 메모리 주소의 내용을 쉘코드로 바꿔주어야 한다.

우선 저장된 return address 전까지 스택의 내용을 채우려면 parse\_url\_subst 함수가 총 263번 호출되어야 한다. 264번째 호출에서 hdr\_buf에 들어갈 내용을 쉘코드로 채우면 성공적으로 쉘을 띄울 수 있다. 다음은 위 사실들을 기반으로 작성한 취약점의 Poc code이다.

1. parse\_url\_subst 함수는 @@ARCH@@ 와 같은 문자열을 현재 시스템의 아키텍쳐 이름 등으로 치환해주는 역할을 하고 있다. 따라서 @@XXXX@@ 형식의 정해진 특정 문자열(parse.c 파일에 주석으로 정리되어 있다.)이 아닌 경우 치환이 적용되지 않는다. [↑](#footnote-ref-1)