

Projeto de programação 2

Cibersegurança - 2023/1 - Prof. Michele Nogueira

Universidade Federal de Minas Ferais

Luís Felipe Ramos Ferreira - 2019022553

Link para o repositório

- Parte 1 - *Exploits* dos alvos 1 e 2

A parte 1 do projeto de programação 2 consistiu em escrever *exploits* que tirassem proveito das vulnerabilidades presentes nos alvos 1 e 2. Em particular, os alvos em questão são códigos escritos em C que contêm vulnerabilidades devido à possibilidade de causar um *buffer overflow*.

No alvo 1, a função *strcpy* é utilizada para armazenar o conteúdo da variável *arg* em *out*. Como *strcpy* não faz nenhuma checagem de limite de tamanho de *string* de cópia, o que permite um *overflow* no momento da cópia. Essa vulnerabilidade permite que um *shellcode* seja introduzido como entrada para o *script* e, assim, seja possível ter acesso à um *shell root*.

No alvo 2, o problema está na implementação da função auxiliar *nmemcpy*. Mais especificamente, na função que exerce a cópia do conteúdo de uma *string* em outra, apesar de existir uma checagem de tamanho das *strings* de cópia, o laço *for* possui um *typo*. A condição de limite imposta no laço é $i \leq \text{len}$ e não $i < \text{len}$. Desse modo, um *byte* a mais é sempre lido na cópia de uma *string* em outra, o que, mais uma vez, permite que o *buffer overflow* seja explorado e um *shellcode* que concede acessos privilegiados a um *shell root* seja possível.

- Parte 2 - Alvos 3, 4, e 5

O alvo 3 parece conter mais um caso de vulnerabilidade devido a *overflow*, mas dessa vez relacionado a números inteiros, que pode ser explorado na função *foo*. Em particular, na função, há uma checagem, antes da cópia, se a variável *count* é menor do que a constante definida *MAX_WIDGETS*. No entanto, o valor de *count* pode ser manipulado para que essa condição seja aceita mesmo quando não deveria. Uma estratégia seria utilizar dos conceitos de tipos de inteiros com e sem sinais em C para alterar o tamanho do *buffer* e assim inserir o *shellcode* no programa.

O alvo 4 sofre da mesma vulnerabilidade do alvo 2. O *typo* presente no laço *for* feito na função auxiliar de cópia de *strings* permite que um *byte* extra seja adicionado na cópia. Ademais, a manipulação dos ponteiros da função *foo*, após o *buffer overflow*, pode permitir que o ponteiro de execução seja direcionado para o *shellcode* já citado.

Para explorar o alvo 5, utilizei o *script find_gadgets.py* para explorar os

gadgets em potencial que permitiriam explorar o *shellcode*. Os resultados foram salvos em um arquivo para análise, mas não obtive sucesso em encontrar a vulnerabilidade para por em prática. No entanto, na teoria, a ideia se resume em encontrar a posição de memória vulnerável que permite manipular o código e executar o *shellcode*.

- Parte 3 - Vulnerabilidades em um programa do mundo real (bdstar)

Após utilizar o *GDB*, o resultado da análise dos dois *crashes* encontrados foram os seguintes:

Podemos ver que ambas aconteceram no arquivo *libarchive/archive_read_support_filter_compress.c* a falha ocorrida foi de *Segmentation fault*, que ocorreu provavelmente ao tentar dereferenciar o ponteiro *state*, que provavelmente não é acessível, e por isso o erro e a consequente vulnerabilidade.

```
(gdb) run -O -xf results/crashes/ld:000000,sig:11,src:000000,time:934788,op:fltp1,pos:4
Starting program: /home/user/proj1/fuzz/install/bin/bdstar -O -xf results/crashes/ld:000000,sig:11,src:000000,time:934788,op:fltp1,pos:4
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
bsdtar: Error opening archive: Failed to open 'results/crashes/ld:000000,sig:11,src:000000,time:934788,op:fltp1,pos:4'
[Inferior 1 (process 861) exited with code 01]
(gdb) run -O -xf results/crashes/
README.txt
ld:000000,sig:11,src:000045,time:705093,op:havoc,rep:2
ld:000001,sig:11,src:000168,time:2734183,op:havoc,rep:8
5093,op:havoc,rep:2
Starting program: /home/user/proj1/fuzz/install/bin/bdstar -O -xf results/crashes/ld:000000,sig:11,src:000045,time:705093,op:havoc,rep:2
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Program received signal SIGSEGV, Segmentation fault.
0x0000000000440b1a in next_code (self=self@entry=0xe310b0)
    at libarchive/archive_read_support_filter_compress.c:386
386      state->stackp++ = state->suffix_code;
(gdb) █
```

Figure 1: *Crash 1*

```
avoc,rep:8-O -xf results/crashes/ld:000001,sig:11,src:000168,time:2734183,op:havoc,rep:8
Starting program: /home/user/proj1/fuzz/install/bin/bdstar -O -xf results/crashes/ld:000001,sig:11,src:000168,time:2734183,op:havoc,rep:8
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Program received signal SIGSEGV, Segmentation fault.
0x0000000000440b1a in next_code (self=self@entry=0xe310b0)
    at libarchive/archive_read_support_filter_compress.c:386
386      state->stackp++ = state->suffix_code;
(gdb) █
```

Figure 2: *Crash 2*

O *backtrace* do *GDB* pode ser visto abaixo:

O *link* com redirecionamento para o repositório público no *GitHub* está disposto nos *headers* deste arquivo.

```

(gdb) bt
#0 0x00000000446b1a in next_code (self=self@entry=0xe310b0)
    at libarchive/archive_read_support_filter_compress.c:386
#1 0x00000000446698 in compress_filter_read (self=0xe310b0, pblock=0xe31138)
    at libarchive/archive_read_support_filter_compress.c:287
#2 0x00000000443d54 in __archive_read_filter_ahead (filter=0xe310b0,
    min=<optimized out>, avail=<optimized out>)
    at libarchive/archive_read.c:1275
#3 0x00000000445fb4 in bzip2_reader_bld (self=self@entry=0xe14c28,
    filter=0x5315f0) at libarchive/archive_read_support_filter_bzip2.c:134
#4 0x00000000443af98 in choose_filters (a=0xe14b20)
    at libarchive/archive_read.c:562
#5 archive_read_open1 (a=_a@entry=0xe14b20) at libarchive/archive_read.c:506
#6 0x00000000444d19 in archive_read_open_filenames (a=<optimized out>,
    a@entry=0xe14b20, filenames=<optimized out>,
    filenames@entry=0x7fffffff09a0, block_size=<optimized out>,
    block_size@entry=10240) at libarchive/archive_read_open_filename.c:150
#7 0x00000000444b47 in archive_read_open_filename (a=0x5315f0,
    a@entry=0xe14b20,
    filename=0x7fffffff76b "results/crashes/id:000001,sig:11,src:000168,time:2734183,op:havoc,rep:8", block_size=15278080)
    at libarchive/archive_read_open_filename.c:107
#8 0x00000000442a734 in read_archive (bsdtar=bsdtar@entry=0x7fffffff200,
    mode=mode@entry=120 'x', writer=writer@entry=0xe13620) at tar/read.c:204
--Type <RET> for more, q to quit, c to continue without paging--
#9 0x0000000042b2c7 in tar_mode_x (bsdtar=bsdtar@entry=0x7fffffff200)
    at tar/read.c:104
#10 0x000000004289f7 in main (argc=<optimized out>, argv=<optimized out>)
    at tar/bsdtar.c:804

```

Figure 3: *Backtrace*