Universidade do Minho
Escola de Engenharia

UNIVERSIDADE DO MINHO

Guião – Aula 10

VULNERABILIDADE DE INTEIROS

Autores:

Bruno Rodrigues: pg41066

Carlos Alves pg41840

QUESTÃO 1.1

Analise o programa underflow.c.

QUESTÃO 1.1.1

Qual a vulnerabilidade que existe na função vulneravel() e quais os efeitos da mesma?

A vulnerabilidade consiste nos números inteiros inicialmente designados no método(x e y), pois estes são do tipo size_t, um inteiro unsigned e são usados para representar o tamanho dos objetos em bytes, em principio o tamanho máximo dependerá do compilador, se for de 32 bits é só um typedef – int sem sinal. Além do problema do tamanho é deveras importante garantir que os valores sejam não negativos (loop infinito e segmentação falsificada).

Mas nesta função diz respeito ao tamanho designado para a matriz(alocação de memoria). Com isto podemos afirmar que se os valores x e y for superior ao valor suportado pela variável i, então a variável entra no loop infinito. Nesta função isto acontece duas vezes. Além disto se observarmos a multiplicação das variáveis do tipo size_t, podemos verificar a possibilidade de um dos problemas referidos no primeiro paragrafo (segmentação falsificada).

QUESTÃO 1.1.2

Complete o main() de modo a demonstrar essa vulnerabilidade.

Para demonstrar esta vulnerabilidade foi necessário modificar um pouco a main(), como se pode verificar no print screen abaixo.

```
int main() {
    char *coiso = 'coisinho';
    //vulneravel(coiso, 0);
    //x superior a 2147483789
    vulneravel(coiso,2147483789,12312312,1)
```

QUESTÃO 1.1.3

Ao executar dá algum erro? Qual?

Sim, Segmentation Fault.

QUESTÃO 1.2

Analise o programa underflow.c.

QUESTÃO 1.2.1

Qual a vulnerabilidade que existe na função vulneravel() e quais os efeitos da mesma?

A vulnerabilidade consiste essencialmente no tamanho designado no método (size_t tamanho), ou melhor como não temos nenhuma forma de garantir que a variável (size_t *tamanho)* seja 0, neste caso e após realizar a operação tamanho – 1 com o valor 0, iriamos verificar um underflow. Pois dados deste tipo não tem qualquer representação negativa.

QUESTÃO 1.2.2

Complete o main() de modo a demonstrar essa vulnerabilidade.

De modo apresentar esta vulnerabilidade, basta chamar a função com *size_t tamanho sendo* 0.

QUESTÃO 1.2.3

Ao executar dá algum erro? Qual?

Sim, verificamos que a função `memcpy` responsável por copiar n caracteres da memoria source para a memória destino faz um esforço em escrever para uma variável de destino que é NULL, e como esperado é reportado o erro Segmentation Fault.

QUESTÃO 1.2.3

Utilize as várias técnicas de programação defensiva introduzidas na aula teórica para mitigar as vulnerabilidades.

Como já referido na primeira questão desta vulnerabilidade, a solução consistiria na implementação de uma verificação do input e claro de possíveis underflows.

```
void vulneravel (char *origem, size_t tamanho) {
        size_t tamanho_real;
        char *destino;
        //tamanho menor que tamanho maximo suportado e maior que 1
        if (tamanho < MAX_SIZE && tamanho > 1) {
                tamanho_real = tamanho - 1; // Não copiar \0 de origem para destino
                destino = (char *) malloc(tamanho_real);
```