Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет безопасности информационных технологий

Дисциплина:

«Операционные системы»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

«Membomb»

Выполнил:

Суханкулиев Мухаммет,

студент группы N3246

(подпись)

Проверил:

Савков Сергей Витальевич,

инженер

(отметка о выполнении)

(подпись)

Санкт-Петербург

2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы – изучение и демонстрация воздействия Membomb на операционную систему.

Для достижения поставленной цели необходимо решить следующие задачи:

- Написать программу выделения памяти и заполнения ее нулями с шагом, равным размеру страницы памяти (mmap, VirtualAlloc);
- Составить график свободной памяти;
- Ознакомиться с работой демона OOM Killer в Linux;
- Достичь сообщения о невозможности выделить память в Windows.

Membomb (Memory Bomb) — это программа, которая намеренно создает чрезмерную нагрузку на оперативную память системы, выделяя большое количество памяти до тех пор, пока она не исчерпается.

1 MEMBOMB ДЛЯ LINUX

1.1 Задание

1.1.1 Написать программу membomb для Linux

1.1.2 Составить график свободной памяти

1.1.3 Ознакомиться с работой демона OOM Killer в Linux

1.2 Выполнение задания

Запускаем bash-скрипт monitoring, который будет работать во время выполнения membomb и запишет в файл memory_log.csv данные свободной памяти. После этого выполнением graph.py составляем график свободной памяти с промежутком в 2 секунды.

P.s. в membomb вставил ограничение, ибо очень долго ждать, пока OOM Killer сам запустится.

1.2.1 Исходные коды

membomb.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <unistd.h>
#define BOMB
int main() {
    long mempagesize = sysconf(_SC_PAGESIZE);
    int pages_allocated = 0;
#ifdef BOMB
    while (1) {
        char *mem = mmap(NULL, mempagesize, PROT_READ | PROT_WRITE,
MAP_PRIVATE | MAP_ANONYMOUS, -1, 0);
        if (mem == MAP_FAILED) {
            printf("Memory allocation failed after %d pages\n",
pages_allocated);
            break;
        }
        pages_allocated++;
        for (long i = 0; i < mempagesize; i++) {
            mem[i] = 0;
        }
    }
#endif
    return 0;
}
```

monitoring

```
#!/bin/bash
echo "Time,FreeMemory(kB)" > memory_log.csv
./membomb &
```

```
BOMB_PID=$!
while kill -0 $BOMB_PID 2> /dev/null; do
    free_mem=$(free | grep Mem | awk '{print $4}')
    echo "$(date +%s),$free_mem" >> memory_log.csv
    sleep 2
done
```

graph.py

```
import matplotlib.pyplot as plt
import csv
import datetime
times = []
free_memory = []
with open('memory_log.csv', 'r') as file:
    reader = csv.reader(file)
    next(reader)
    for row in reader:
        timestamp = int(row[0])
        memory = int(row[1])
        times.append(datetime.datetime.fromtimestamp(timestamp))
        free_memory.append(memory)
plt.figure(figsize=(10, 6))
plt.plot(times, free_memory, label="Free Memory (kB)", color="blue",
linewidth=2)
plt.xlabel('Time')
plt.ylabel('Free Memory (kB)')
plt.title('Free Memory Over Time During Membomb Execution')
plt.grid(True)
plt.legend()
plt.show()
```

1.3 Скриншоты выполнения

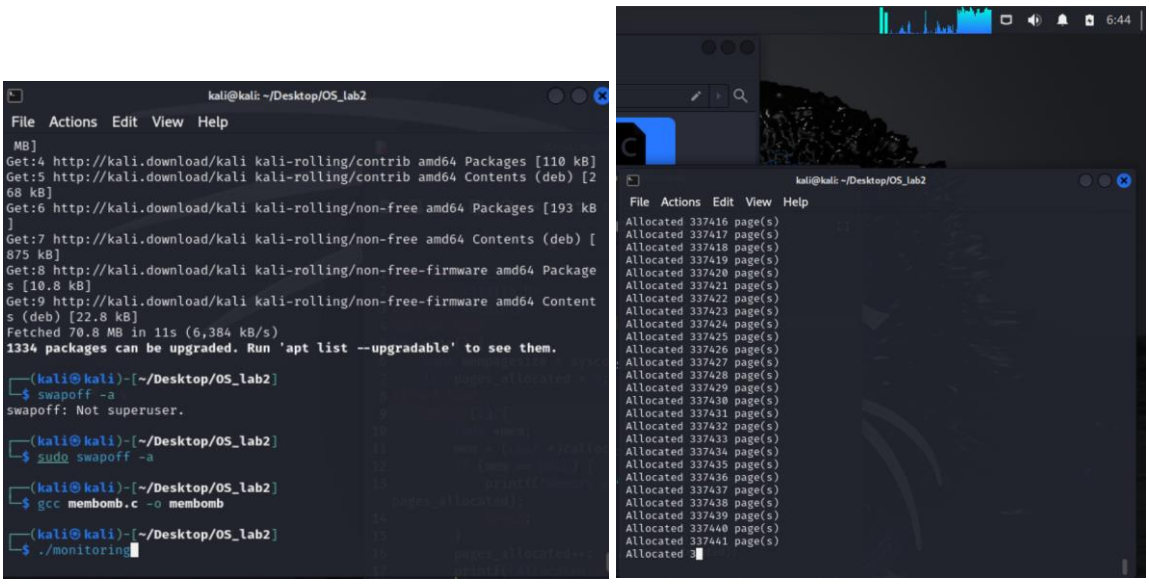

Рисунок 1 – Выполнение программы

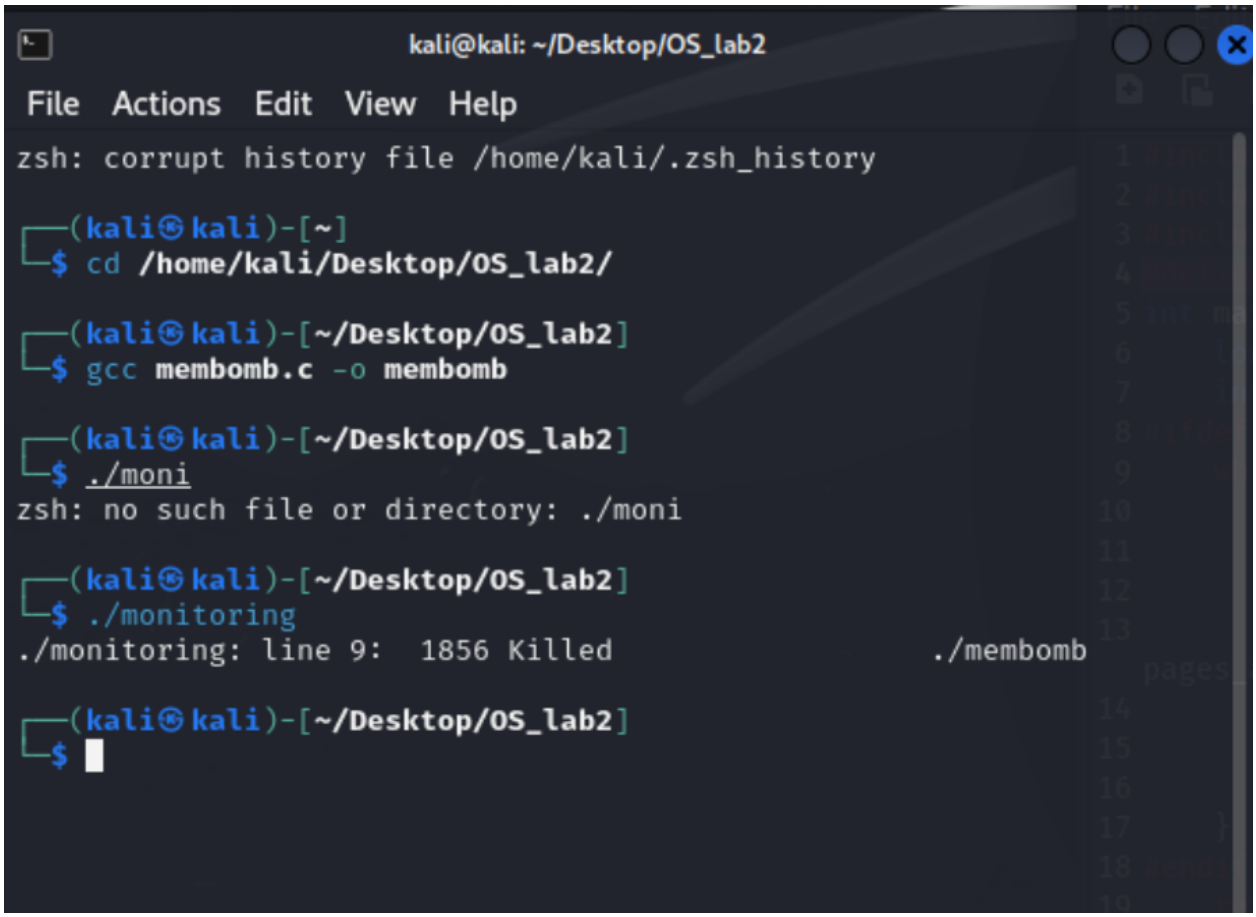

Рисунок 2 – OOM Killer (Убрал printf в membomb)

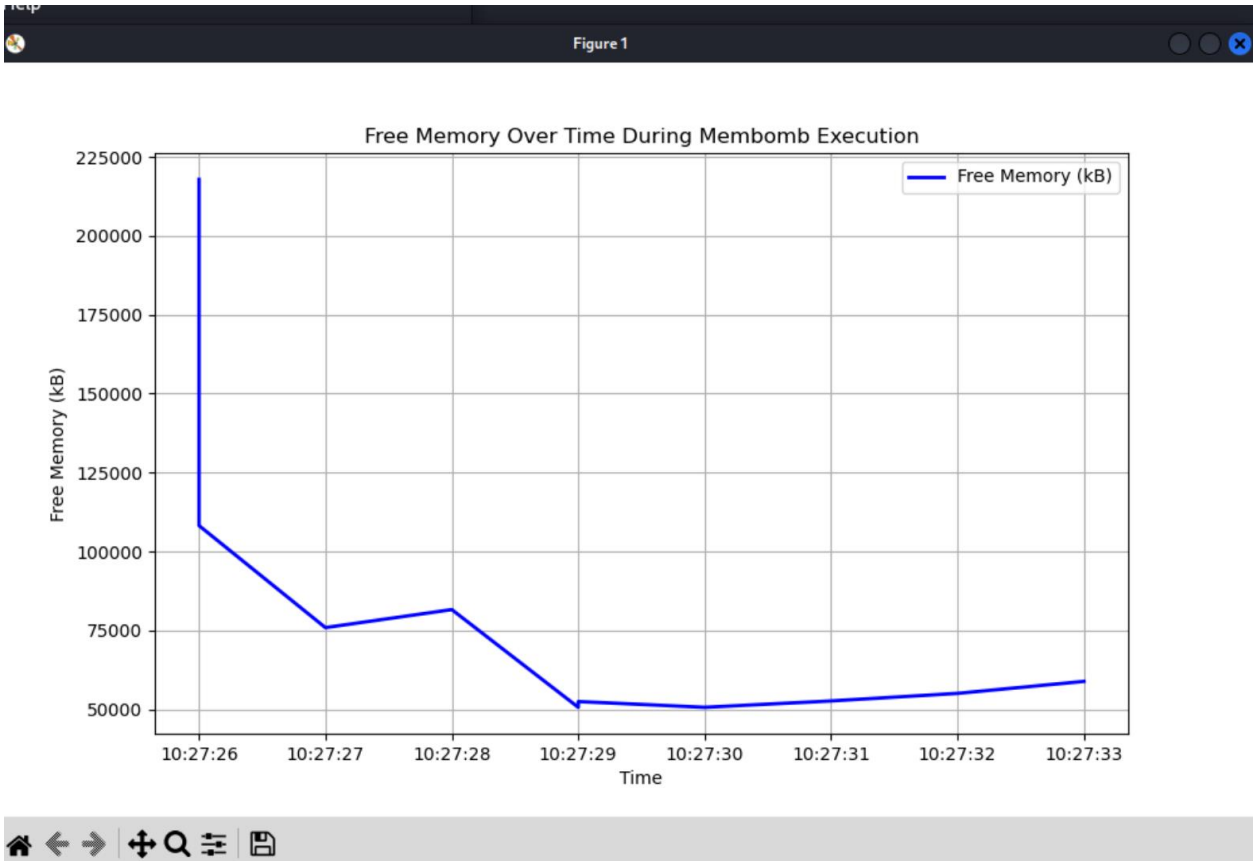

Рисунок 3 – График свободной памяти

1.4 Подробнее

OOM Killer (Out of Memory Killer) — это механизм в Linux, который срабатывает, когда система сталкивается с нехваткой оперативной памяти. Когда памяти не хватает, OOM Killer выбирает процессы с наибольшим потреблением ресурсов или те, которые наименее критичны для работы системы, и завершает их, чтобы освободить память и предотвратить зависание системы.

2 MEMBOMB ДЛЯ WINDOWS

2.1 Задание

2.1.1 Написать программу membomb для Windows

2.1.2 Составить график свободной памяти

2.1.3 Достичь сообщения о невозможности выделить память в Windows

2.2 Выполнение задания

Запускаем monitoring.exe, который будет работать во время выполнения membomb.exe и запишет в файл memory_log.csv данные свободной памяти. После этого выполнением graph.exe составляем график свободной памяти с промежутком в 2 секунды.

2.2.1 Исходные коды

membomb.py

```
import ctypes
PAGE_SIZE = 4096
allocated_memory = []
pages_allocated = 0
try:
    while True:
        # Выделяем память
        mem = ctypes.windll.kernel32.VirtualAlloc(
            None,
            PAGE_SIZE,
            0x1000 | 0x2000,  # MEM_COMMIT | MEM_RESERVE
            0x04  # PAGE_READWRITE
        )

        if mem == 0:
            print(f"Memory allocation failed after {pages_allocated} pages.")
            break
        allocated_memory.append(mem)
        pages_allocated += 1
        print(f"Allocated {pages_allocated} page(s)")
except KeyboardInterrupt:
    print("Program interrupted.")
for mem in allocated_memory:
    ctypes.windll.kernel32.VirtualFree(mem, 0, 0x8000)  # FREEMEM
```

monitoring.py

```
import psutil
import csv
import time
with open('memory_log.csv', mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(['Time', 'FreeMemory(MB)'])
    try:
        while True:
```

```
            free_memory = psutil.virtual_memory().available / (1024 * 1024)
            writer.writerow([time.strftime('%Y-%m-%d %H:%M:%S'),
free_memory])
            time.sleep(1)
    except KeyboardInterrupt:
        print("Logging stopped.")
```

graph.py

```
import matplotlib.pyplot as plt
import pandas as pd
data = pd.read_csv('memory_log.csv')
data['Time'] = pd.to_datetime(data['Time'])
plt.figure(figsize=(10, 5))
plt.plot(data['Time'], data['FreeMemory(MB)'], marker='o', linestyle='-')
plt.title('Free Memory Over Time')
plt.xlabel('Time')
plt.ylabel('Free Memory (MB)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.grid()
plt.savefig('memory_graph.png')
plt.show()
```

2.3 Скриншоты выполнения

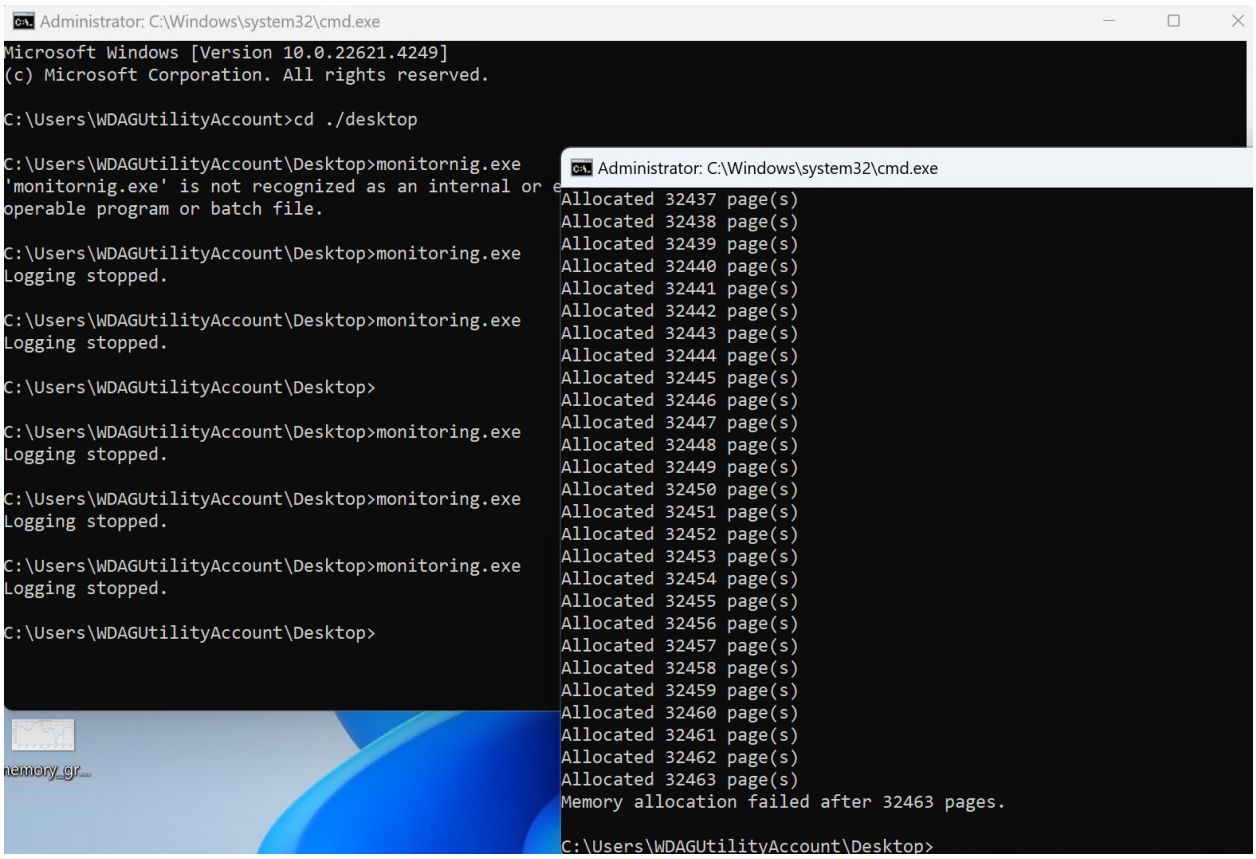

Рисунок 4 – Выполнение программы

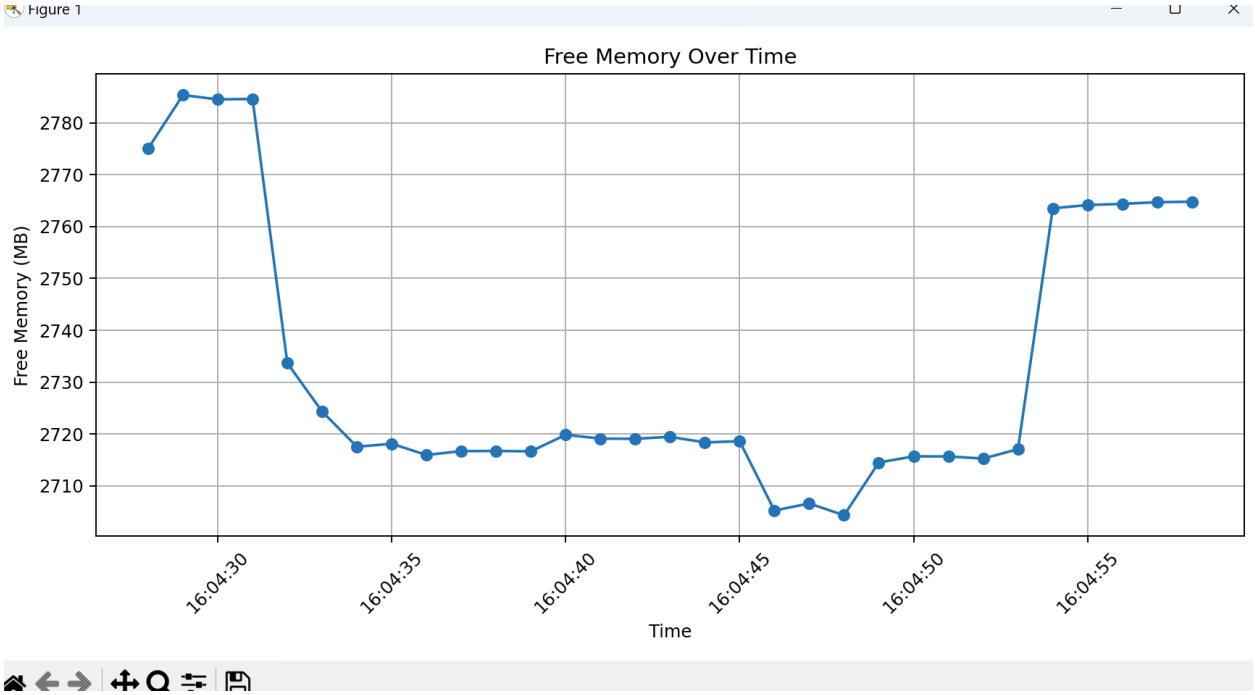

Рисунок 5 – График свободной памяти

ЗАКЛЮЧЕНИЕ

В ходе работы была успешно разработана и протестирована программа Membomb для Windows и Linux, демонстрирующая влияние чрезмерного выделения памяти на операционную систему. Также были составлены графики свободной памяти, наглядно показывающие, как система реагирует на выделение ресурсов. Ознакомление с механизмом OOM Killer в Linux позволило понять, как операционная система управляет памятью и предотвращает зависание.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. GitHub - GlitchPunkWTF/membomb: membomb
2. Презентация PowerPoint (sibsutis.ru)
3. Linux Kernel Documentation: Out of Memory (OOM) Killer — Описание механизма OOM Killer и его работы в ядре Linux.