RV College of Engineering®

Experiential Learning Phase -I :
CS235AI
Operating Systems

DETECTING MEMORY ABNORMALITIES USING MEMORY SCANNER

by
Sohan Varier (1RV22CS200)
Vaibhav Soin (1RV22CS221)

Go, change the world®

RV College of Engineering®

PRESENTATION CONTENTS

Go, change the world®

- Problem Statement
- Relevance in Course
- Methodology
- Literature Survey
- Tools/APIs used
- Application of the project
- Partial Implementation

Problem Statement

The objective of this project is to develop a memory scanning tool capable of detecting various types of memory abnormalities in computer systems. Memory abnormalities can lead to system crashes, data corruption, and other critical issues, making it essential to identify and address them promptly. The memory scanner should be able to analyze the system's memory and identify potential problems such as memory leaks, buffer overflows, invalid memory accesses, and other memory-related vulnerabilities.

Relevance of Project in Current Course

- The Operating Systems course in our curriculum covers memory management and virtual memory of applications
- Our project aims to make use of APIs available through C to access the memory of currently running applications
- Using these APIs we can peek and at particular memory locations, and also check for various abnormalities

Methodology

CREATE A MEMORY SCANNER TO MONITOR THE PROCESS WHILE RUNNING

RESEARCH MEMORY ABNORMALITIES AND HOW TO CHECK FOR THEM

DEVELOP DETECTION ALGORITHMS TO IDENTIFY MEMORY ABNORMALITIES BASED ON MEMORY ANALYSIS

FUTURE SCOPE: DETECTING MEMORY ABNORMALITIES EVEN WHEN PROCESSES ARE IDLE AND ACTIVELY FIX THEM

Tools/APIs Used

- <windows.h> C library
- VirtualQueryEx: Used to retrieve information about a range of pages within the address space of a specified process.
- ReadProcessMemory: Reads data from the specified process's memory.
- WriteProcessMemory: Writes data to the specified process's memory.

Working of the Memory Scanner

- The memory scanner takes 3 primary inputs: PID of process to scan, size of data to scan, and initial value of data
- Using this information, it creates a linked-list of memory blocks of the memory of the process satisfying these criteria
- Initially, there may be multiple instances of the similar memory content, so we iteratively enter the updated values of the data that we want to find
- Finally it, you will be able to peek into the memory of the app, and check for abnormalities

Working of the Memory Scanner

Structure of linked list “MEMBLOCK”:

HANDLE hProc	Represents the handle to the process
unsigned char *addr	Points to the base address of the memory block
int size	Size of memory blocks in bytes
unsigned char *buffer	Buffer to copy data into while reading/manipulating
unsigned char *searchmask	Search mask for tracking matching conditions
int matches	Number of matches in the memory block
int data_sizes	Size of data elements in the memory block
struct _MEMBLOCK *next	Pointer to next memory block

Applications

- Identifying and fixing memory leaks in software applications, which can lead to resource depletion and performance issues.
- Detecting buffer overflows and other memory-related vulnerabilities, which can be exploited for security breaches.
- Assisting in the debugging process by pinpointing memory-related issues during application development and testing.
- Monitoring system memory usage and detecting potential memory-related issues in servers, workstations, and other critical systems.
- Conducting security audits and penetration testing by identifying memory-related vulnerabilities that could be exploited by attackers.
- Analyzing the memory footprint of applications and systems to detect potential memory-related attack vectors.
- Detecting memory abnormalities in resource-constrained embedded devices and IoT sensors, where memory leaks or buffer overflows can have severe consequences.

Summary of the Phase I report

Phase 1 of our project has concluded with the successful development of the MemBlock memory scanner module, facilitating peek operations within targeted memory spaces. All APIs have been tested, ensuring proper functionality. Next steps involve implementing abnormality detecting operations, refining existing functionalities, optimizing performance. Documentation and reporting will be maintained throughout the development process. Phase 1's achievements lay a solid foundation for the project's progression, positioning us to enhance the capabilities and effectiveness of our memory scanning tool in subsequent phases.

Various Memory Abnormalities

Memory abnormalities, in the context of computer systems, refer to unexpected or anomalous behaviors related to memory usage or memory management. These abnormalities can arise from various factors, including software bugs, security vulnerabilities, hardware failures, or malicious activities. Detecting and addressing memory abnormalities is crucial for maintaining system stability, security, and reliability.

Examples:

1. Memory Leaks: occur when a program allocates memory dynamically but fails to release it after it's no longer needed.
2. Buffer Overflows: occur when a program writes data beyond the bounds of a buffer, leading to corruption of adjacent memory locations.
3. Code Injection: involves inserting and executing malicious code into a running process's memory space.
4. Heap corruption: it is the unintended modifications of memory allocated on the heap

1. Enhancing OS Memory Management Performance:Authors:Nihad ramadhan omar, Dohuk Polytechnic University
2. Sternberg, Saul. (1975). Memory Scanning: New Findings and Current Controversies. Quarterly Journal of Experimental Psychology - QUART J EXP PSYCHOL. 27. 1-32. 10.1080/14640747508400459.
3. Hamdioui, Said & Al-Ars, Zaid. (2009). Scan more with memory scan test. 204 - 209. 10.1109/DTIS.2009.4938056.
4. Memory Management in Operating System" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-2 Issue-5, August 2018