MEMORY SCANNER AND ANOMALY DETECTION

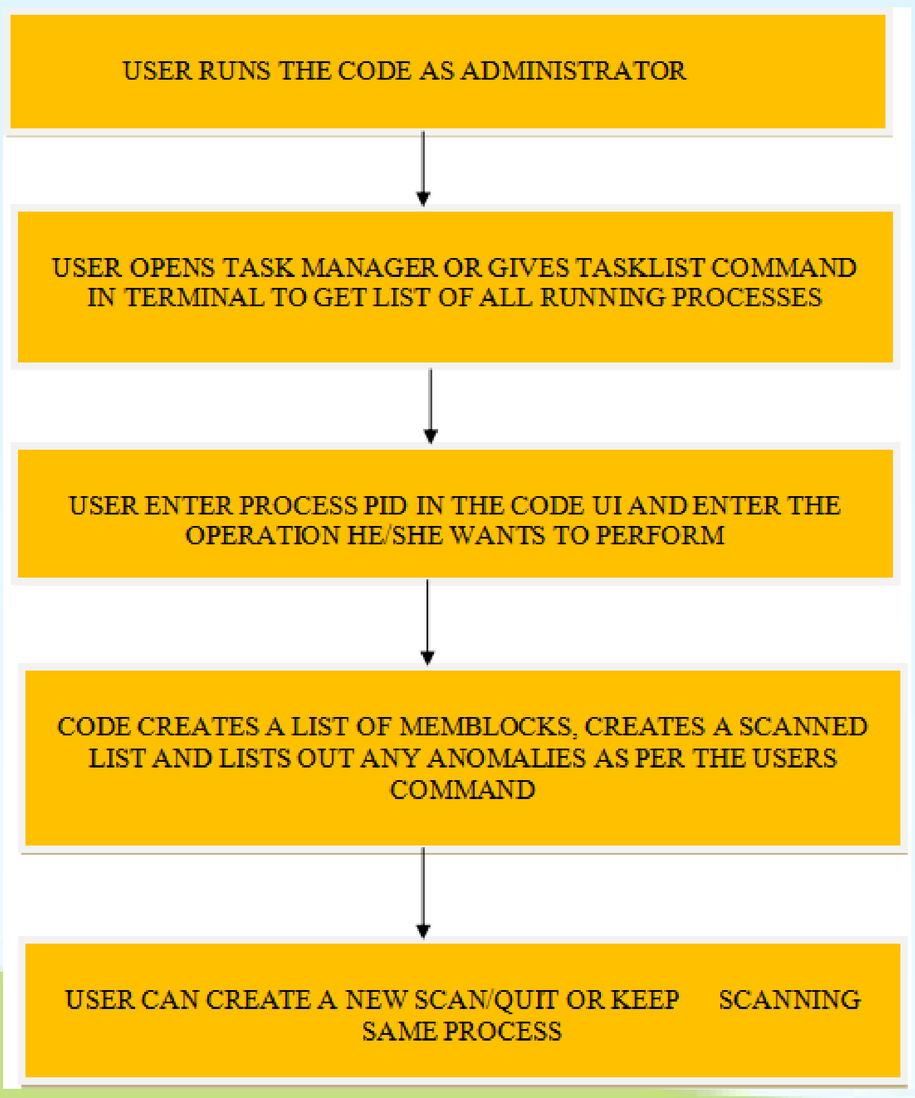
Sohan Varier (1RV22CS200)
Vaibhav Soin(1RV22CS221)

Faculty Guide:
Jyoti Shetty
Associate Professor
Dept. of CSE
RV College of Engineering

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.

METHODOLOGY



MEMBLOCK STRUCTURE

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

PROJECT OVERVIEW

PART 1:
Creating a memory scanner which can scan through an entire process and return values stored in each data block. Using this, we can locate an memory block in the process and modify its value. Furthermore, we can also scan for memory blocks with values greater or lesser than a reference value and dynamically update the list.
PART 2:
The second part of our project involves using the memory scanner built in part one to our advantage. We scan a process for anomalies and return warnings if any are found. We have extended to scanner to detect 3 anomalies as of now. They are:
1)Code Injection.
2)Buffer Overflow
3)DLL Injection

SYSTEM CALLS USED

The system calls used in the code are:
1. VirtualQueryEx
2. ReadProcessMemory
3. WriteProcessMemory
4. VirtualProtectEx
5. OpenProcess
6. CloseHandle

OUTPUTS

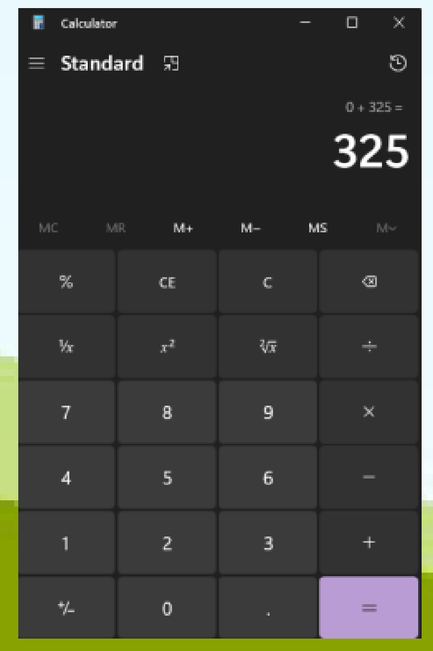
```
5650055 matches found

Enter the next value or
[i] increased
[d] decreased
[m] print matches
[j] check for memory injection
[b] check for buffer overflows
[h] check for heap correction
[n] new scan
[q] quit
325

3 matches found

Enter the next value or
[i] increased
[d] decreased
[m] print matches
[j] check for memory injection
[b] check for buffer overflows
[h] check for heap correction
[n] new scan
[q] quit
m

0x302cec00: 0x00000145 (325)
0x30225fa0: 0x00000145 (325)
```




```
Enter the next value or
[i] increased
[d] decreased
[m] print matches
[j] check for memory injection
[b] check for buffer overflows
[h] check for heap correction
[n] new scan
[q] quit
b

0 buffer overflows found.

Enter the next value or
[i] increased
[d] decreased
[m] print matches
[j] check for memory injection
[b] check for buffer overflows
[h] check for heap correction
[n] new scan
[q] quit
h

Heap corruption not detected
```

```
Enter the next value or
[i] increased
[d] decreased
[m] print matches
[j] check for memory injection
[b] check for buffer overflows
[h] check for heap correction
[n] new scan
[q] quit
j

Potential code injection detected at address 0x00007ff675067000
Potential code injection detected at address 0x00007ff8f7be3000
Potential code injection detected at address 0x00007ff904881000
Potential code injection detected at address 0x00007ff9052b1000
Potential code injection detected at address 0x00007ff906080000
Potential code injection detected at address 0x00007ff906082000
Potential code injection detected at address 0x00007ff906084000
Potential code injection detected at address 0x00007ff906086000
Potential code injection detected at address 0x00007ff906088000
Potential code injection detected at address 0x00007ff90608a000
Potential code injection detected at address 0x00007ff90608c000
Potential code injection detected at address 0x00007ff90608e000
Potential code injection detected at address 0x00007ff9060aa000
Potential code injection detected at address 0x00007ff906769000
```