NAME: _____

Reverse Engineering FINAL EXAM Review

Spring 2023
Computer Science Department, New Mexico State University

1 Review Topics

You can expect 20 multiple choice questions on the tools and skills we have used in the course, and 30 practical questions based on skills from the class projects.

1. You don't need to know everything about each of the following. You do need to know what each of these is, how we've used them, and why. The questions I ask will fit into these categories.
    1. The stack
    2. Machine code
    3. Assembly code
    4. Basic assembly instructions like mov, sub, call, ret, leave, push, pop, etc.
    5. Common registers in 32-bit x86 (esp, ebp, eax, ebx, esp, eip)
    6. Common registers in 64-bit x86 (rsp, rbp, rax, rbx, rsp, rip)
    7. Stack frames
    8. Buffer Overflow
    9. Shellcode
    10. System Calls
    11. DLLs
    12. DllMain()
    13. DLL injection
    14. LoadLibraryA
    15. PIDs
    16. Ransomware
    17. Control flow graphs
    18. Control flow sources and sinks
    19. Our ideas where to start with RE (Find source/sink, Look for known library functions, work backwards)
    20. GDB
    21. ltrace, strace, uftrace
    22. pwntools
    23. core dump files
2. This exam will focus on the material from the second half of the course, but there will be some questions where knowing things from the first part of the semester will be required. For your convenience, the midterm review outline is included here. You should know what each of the following tools do, and why they are useful to a reverse engineer

1. InetSim
2. Hypervisors
3. Virtual machines
4. Wireshark
5. Procmon
6. regshot
7. strings
8. PE Explorer
9. DependencyWalker
10. virtual machine snapshots
11. sandbox services
12. VirusTotal
13. PEiD
14. Ghidra

3. You should also know about the following concepts that are not specifically tools:
    1. network isolation
    2. process isolation
    3. ransomware
    4. viruses vs worms
    5. packers
    6. indicators of compromise
    7. common registers and instructions in x86
    8. static vs dynamic analysis

2 Reverse Engineering

You will be shown snapshots similar to weekly projects we completed throughout the semester. Similar to the midterm's applied questions, the final will test if you understand what is happening in these snippets. We'll do some sample questions in the last class to allow you to test yourself.

4. Crackmes

```
int main(undefined param1, undefined ** param2) {

    char * sVar1 = NULL;
    undefined uVar1 = 0;
    FUNCTION_00493828(s_Please_Input_Password_00432A1C, &sVar1, &uVar1);

    if ((strcmp(sVar1, s_NMSU_RE_Forever) != 0)) {
        printf(s_Wrong_Password_00432AEE);
        return 1;
    } else {
        printf(s_Correct_Password_00432B3A);
        return 0;
    }
}
```

5. DLL Main

```
int main(int iVar1, undefined* uVar1[]) {
   HANDLE hVar1;
   undefined uVar2;
   char* strVar1 = "C:\\NMSU-RE\\evil.dll";
   hVar1 = OpenProcess(0x1fffff, 0, DWORD(atoi(uVar[1])));
   uVar2 = VirtualAllocEx(hVar1, NULL, sizeof strVar1, MEM_COMMIT, PAGE_READWRITE);
   WriteProcessMemory(hVar1, uVar2, strVar1, sizeof strVar1, NULL);
   CreateRemoteThread(
        hVar1,
        NULL,
        0,
        GetProcAddress(GetModuleHandle(TEXT("Kernel32")), "LoadLibraryA"),
        uVar2,
        0,
        NULL
    );
   return 0;
}
```

6. Ransomware

```
void decrypt(char * param1, char * param2) {
    FILE *fptr1;
    FILE *fptr2;
    char cArr1[1];
    long lVar1;
    char cVar1 = '4';

    fptr1 = fopen(param1, "rb");
    fptr2 = fopen(param2, "wb");
    fseek(fptr1, 0, SEEK_END);
    lVar1 = ftell(fptr1);
    rewind(fptr1);

    for(int iVar1 = 0; iVar1 < lVar1; iVar1++) {
        fread(cArr1, 1, 1, fptr1);
        *(cArr1) ^= cVar1;
        fwrite(cArr1, 1, 1, fptr2);
    }
    return;
}
```

7. Shellcode

```
    pushq $0
    movq %rsp, %rdx
    push %rdx
    movq %rsp, %rsi
    movq %rsp, %rdx
    movq $0x0068732F6E69622F, %rax
    pushq %rax
    movq %rsp, %rdi
    movq $0x3B, %rax
    movq $0x3B, %rbx
    syscall
```

8. Buffer Overflow

Address	Value	Comment
0x07ff4523AA80	0x00A000FF00000000	
0x07ff4523AA80	0x00F0006800000330	
0x07ff4523AA88	0x4141414100000000	<== RSP
0x07ff4523AA90	0x4141414141414141	
0x07ff4523AA98	0x4141414141414141	
0x07ff4523AAA0	0x4141414141414141	
0x07ff4523AAA8	0x0041414141414141	
0x07ff4523AAB0	0x000007ff4523AAD0	<== RBP
0x07ff4523AAB8	0x000000000040F45A	
0x07ff4523AAB8	0x000007ff4523AAC8	
0x07ff4523AAB8	0x000007ff4523AB00	