2024 NTU Virtual Machine HW4 Writeup

StudentID : R12922054

Idea & Concepts

In this assignment, the objective is to execute a program named *hijack* within the virtual machine (VM). *hijack* attempts to overwrite its own code located in the virtual address range 0x4005e4 to 0x40060b. However, the memory page containing this code is marked as read-only, causing any write attempt to trigger a page fault. This assignment aims to modify the system so that the virtual address range 0x4005e4 to 0x40060b becomes writable, thereby allowing the hijack program to overwrite its code successfully.

The initial approach considered was to modify the TLB entries corresponding to the virtual address range 0x4005e4 to 0x40060b from being marked as read-only to writable.Based on insights from the resource [1], this approach seemed viable. Consequently, the focus shifted to identifying the function responsible for setting TLB attributes. The following sections detail the implementation process.

Implementation

1. First, find this function `arm_cpu_tlb_fill` in `/target/arm/tlb_helper.c`.
2. Second, in the function, `arm_cpu_tlb_fill`, it was observed that the variable, `prot`, determines the permissions for the memory address. Upon completing the execution of the function shown below, the value of `prot` would be set.
```
ret = get_phys_addr(&cpu->env, address, access_type,
                    core_to_arm_mmu_idx(&cpu->env, mmu_idx),
                    &phys_addr, &attrs, &prot, &page_size,
                    &fi, &cacheattrs);
```
3. Third, in the `/target/arm/helper.c` file, within the `get_phys_addr` function, we can see that the `get_phys_addr_lpae` function is executed, shown below. This function determines the `s2_prot` variable, which is then subjected to an `AND` operation with the prot variable. Therefore, we need to delve into the `get_phys_addr_lpae` function to perform operations on the `s2_prot` variable.
```
/* S1 is done. Now do S2 translation.  */

ret = get_phys_addr_lpae(env, ipa, access_type, s2_mmu_idx, is_el0,
                         phys_ptr, attrs, &s2_prot,
                         page_size, fi, &cacheattrs2);
```
4. In the `/target/arm/helper.c` file, within the `get_phys_addr_lpae` function, the following code can be observed:
```
if (!(*prot & (1 << access_type))) {
     goto do_fault;
}
```
If we perform a `memcpy` operation, then `access_type = MMU_DATA_STORE` (where `access_type` has three values: `MMU_DATA_LOAD = 0`, `MMU_DATA_STORE = 1`, and `MMU_INST_FETCH = 2`). However, in the virtual address range `0x4005e4` to `0x40060b`, the memory is read-only, meaning the value of `*prot` is `PAGE_READ`. Therefore, we need to add the following code before the code snippet shown above:
```
if ((address >= 0x4005e4 && address <= 0x40060b){
     *prot |= PAGE_WRITE;
}
```
5. This way, we can successfully avoid entering the `if (!(*prot & (1 << access_type)))` block.

References

[1] "QEMU-TLB," [Online]. Available: https://airbus-seclab.github.io/qemu_blog/tcg_p3.html