

Activity Topic: Shellcode Development

=====

Q1. The essence of a shellcode (32-bit) is to prepare four registers, `eax`, `ebx`, `ecx`, and `edx`, before invoking the `execve()` system call. Describe what values these four registers should contain.

```
eax:
ebx:
ecx:
edx:
```

Q2. In the stack-based approach, we need to store command string in the memory, and then save the string's address in `ebx` register. Write a code snippet (32-bit) to store the string `"aaaabbbbccccdddd"` in the memory, and then save its address to `ebx`.

Q3. In the stack-based approach, we need to store the argument array `argv[]` in the memory, and then store the array's address in `ecx`. Write a code snippet (32-bit) to construct the following `argv[]` array in the memory, and then assign its address to `ecx`.

```
argv[0] = 0x11111111
argv[1] = 0x22222222
argv[2] = 0x33333333
argv[3] = 0x00000000
```

Q4. Why does shellcode in general not allow zeros in the code?

Q5. List three typical solutions to get rid of zeros in shellcode.

Q6. We would like to store a string `"ab"` on the stack, but we are not allowed to include any zero in the code (the end of the string has a binary zero).

Complete the code below (assume the machine is little endian)

```
mov ecx, "ab**"
... (missing code) ...
push ecx
```

Complete the code below (assume the machine is big endian)

```
mov ecx, "ab**"
... (missing code) ...
push ecx
```