C constructs - Assembly x86

Task

The excerpt below is from the code of malware.

```
.text:00401000  push    ebp
.text:00401001  mov     ebp, esp
.text:00401003  push    ecx
.text:00401004  push    0
.text:00401006  push    0
.text:00401008  call    ds:InternetGetConnectedState
.text:0040100E  mov     [ebp+var_4], eax
.text:00401011  cmp     [ebp+var_4], 0
.text:00401015  jz      short loc_40102B
.text:00401017  push    offset aSuccessInternetConnection
.text:0040101C  call    sub_40105F
.text:00401021  add     esp, 4
.text:00401024  mov     eax, 1
.text:00401029  jmp     short loc_40103A
.text:0040102B  ; -------------------------------------------
.text:0040102B
```

1. Identify the Known Constructs

Identify the constructs seen during the theoretical lesson (e.g., while, for, if, switch, etc.).

2. Hypothesize the Functionality

Based on the hint: The function `InternetGetConnectedState` checks if a machine has Internet access. Describe the high-level execution of the code.

3. BONUS: Explain Each Line of Code

Study and explain each individual line of code in detail.

Solution:

Analysis of Assembly Code

1. Identifying Known Constructs

From the given code snippet, we can identify the following known constructs:

- **Function Call (`call`)**: The code calls the `InternetGetConnectedState` function.
- **`if` Control Structure**: Uses `cmp` and `jz` for a conditional jump, checking if the return value of `InternetGetConnectedState` is zero.

2. Hypothesizing the Functionality

The code checks if the machine has Internet access using the `InternetGetConnectedState` function. If the connection is present, the program performs further operations (partially visible in the provided snippet); otherwise, it jumps to another section of the code.

3. BONUS: Explaining Each Line of Code

Here is a line-by-line explanation of the provided assembly code:

```
.text:00401000  push    ebp                   ; Save the value of the base pointer (ebp) on the stack
.text:00401001  mov     ebp, esp              ; Set ebp to the current value of the stack pointer (esp)
.text:00401003  push    ecx                   ; Save the ecx register on the stack
.text:00401004  push    0                     ; Push the value 0 onto the stack (argument for dwReserved)
.text:00401006  push    0                     ; Push the value 0 onto the stack (argument for lpdwFlags)
```

```
.text:00401008  call    ds:InternetGetConnectedState ; Call the InternetGetConnectedState function
.text:0040100E  mov     [ebp+var_4], eax   ; Save the return value of InternetGetConnectedState in [ebp+var_4]
.text:00401011  cmp     [ebp+var_4], 0     ; Compare the value in [ebp+var_4] with 0
.text:00401015  jz      short loc_40102B   ; Jump to loc_40102B if [ebp+var_4] is zero (no connection)
.text:00401017  push    offset aSuccessInternetConnection ; Push the offset of the string "Success: Internet Connection\\n"
.text:0040101C  call    sub_40105F          ; Call the function sub_40105F (probably to print the string)
.text:00401021  add     esp, 4              ; Restore the stack pointer (remove the string from the stack)
.text:00401024  mov     eax, 1              ; Set eax to 1 (probably a success code)
.text:00401029  jmp     short loc_40103A    ; Jump to loc_40103A (end of the check)
.text:0040102B  ; -----------------------------------------------------------
.text:0040102B  ; Code executed if there is no connection (not visible in the provided snippet)
```

Functionality Implemented

The functionality implemented by the assembly code is as follows:

1. **Internet Connection Check**: Uses `InternetGetConnectedState` to determine if the machine has Internet access.
2. **Conditional Execution**:
   - If the connection is present, it prints "Success: Internet Connection\n" and sets a success code (1).
   - If the connection is not present, it jumps to another section of the code (not visible in the provided snippet).

Final Considerations

This code snippet is typical of malware that checks for an Internet connection before performing further operations. By analyzing the conditional behavior and use of API calls, we can infer that the malware might download additional payloads or send data only if an Internet connection is available.

Federico Biggi