* Proven exploitable buffer
* the distance to EIP determined
* sufficient room for shellcode discovered,
* a basic proof-of-concept (POC) exploit was developed.
* the distance to EIP was x
* and the total space for shellcode was 280.
* Required to create shellcode
* Control over EIP gained by filling dist with char
* Post ret in skin loading function
* ESP will point to the start of the exploit shellcode as it is placed directly after the bytes that overwrite EIP.
* The return pops 4 bytes, leaving ESP pointing to the shellcode that follows.
* the value of the ESP will be unknown (unless manually read in a debugger), so a return address shouldn’t be hardcoded.
* There are fixed addresses that contain commands like JMP ESP.
* using one of these addresses guarantees that the payload will consistently begin execution as JMP ESP should jump to the start of the exploit shellcode.
* With that in mind, the EIP is set to an address that contains a JMP ESP.
* With the EIP and padding created the only thing left to add is the shellcode. As this is only a proof of concept, the exploit shellcode used just opens calculator.