You can find examples and explanations of buffer overflow exploits in various projects on the **samsclass.info** website. For instance, Project 3 involves creating a buffer overflow with shellcode on a Linux system. It guides you through compiling code without modern protections, debugging with gdb, overflowing the stack with a long string of characters, and even manipulating the return address to control the execution flow of the program. The example uses gdb to analyze the state of the registers before and after the overflow, and teaches how to inject shellcode into the program's execution. It also explains the concept of a NOP sled, which is a sequence of NOP (No Operation) instructions used to safely land the execution pointer to the desired entry point of the attacker's code.

Another example, Project 8, starts with a simple C program that prompts for a name and then prints a greeting. The program is intentionally written with a buffer overflow vulnerability, which is then exploited by inputting a string longer than the buffer size, causing a segmentation fault. The gdb debugger is used to examine the state of the CPU registers during the crash, showing how the return address can be overwritten by the input, potentially allowing an attacker to take control of the program's execution flow.

These exercises are designed for educational purposes to understand the mechanics behind buffer overflows and the importance of writing secure code. They are part of a larger set of projects aimed at learning exploit development and computer security.

For a complete guide on how to perform these exercises, you should visit the detailed project pages on **samsclass.info**:

* [Project 3: Linux Buffer Overflow With Shellcode](https://samsclass.info/127/proj/p3-lbuf1.htm)
* [Project 8: Buffer Overflow](https://www.samsclass.info/123/proj14/123p8C.htm)

These resources will give you step-by-step instructions on setting up the environment, writing the vulnerable code, and then exploiting it. Please remember to use this information responsibly and ethically, as buffer overflows are a common vulnerability that can lead to serious security breaches.