Both exploit generators may be compiled with a standard gcc command:

```
gcc -o exploit_local exploit_local.c

gcc -o exploit exploit.c
```

To execute the exploits, simply pipe the output of the program into a netcat command with the respective victim IP address and port number:

```
./exploit_local | nc [VICTIM IP ADDRESS] [VICTIM PORT NUMBER]

./exploit | nc 10.247.49.156 8888
```

The exploit for the remote attack on the device located in the infrastructure should return a shell to a device on port **6288**, however, I was unable to get this exploit to function properly. The local exploit works under the conditions that I have tested it. To develop these exploits, I began with a proof-of-concept attack on a locally-run nweb instance on a Redhat9 virtual machine. I started by transferring nweb to the machine using WiNSCP and installing it in the root directory. After installing, I made sure Redhat9 would generate core files of unlimited size (**`ulimit -c unlimited`**) and executed the web server, running it on port **8888**. With the web server running, I then began to attack it using a terminal on a Kali Linux virtual machine. At first, I tried a large number of A's, using the perl script suggested in the assignment guidelines. This attack produced a core file on the victim machine that, when opened using GDB, revealed the EIP had been overwritten by A's (**0x41414141**). To determine where the EIP was in relation to the buffer that had been overflowed, I generated a pattern of length 2000 and, like before, sent it to the web server. This time, the core file on the victim machine revealed an EIP value of **0x42346942** which, when processed by the offset calculator, determined that the offset between the buffer and the EIP was **1032** bytes. With this value, the exploit could now be constructed. Using the Metasploit framework, I generated a payload using the alpha mixed encoder whose listening host (**LHOST**) had been set to the IP address of my Kali Linux virtual machine and the listening port (**LPORT**) had been set to a port of my choosing (**6288** in this case). Combining 1032 NOP instructions, followed by the JMP ESP instruction, followed by 500 more NOPs and, finally, the above payload, I was able to construct an exploit string that, when sent to the running nweb server, returned a shell to the Kali Linux machine who was listening on that port. However, when I tried to substitute the payload for one whose **LHOST** was **10.247.49.156** and **LPORT** was, again, **6288**, I could not seem to get a shell returned on the listening machine. Unfortunately, I did not have enough time to debug my exploit program for this instance, but I suspect it may have been to long to work properly in this context. Had I split the exploit string into two payloads divided by the **JMP ESP** instruction, as it is depicted in the lecture slides, I may have been more successful in this attack.