Phase 1: The buffer is 40 bytes. We want to change the return address in the stack to return to the touch 1 address from GetBuf rather than to the test address. So we fill the buffer with 40 0 characters, then input the address of touch 1 in reverse, which is written from low to high addresses in the stack, but read from high to low addresses, so that the address will be interpreted correctly.

Phase2: This phase is similar, except that we need to move the cookie into %rdi so it will be passed to touch2 as an argument. So we create the instruction to do this in byte code, and input that as the first buffer byte, followed by 4 bytes of zeros to fill the buffer. Next, we input the return address as the address of RSP that holds the cookie instruction, so that the instruction executes and returns, followed by the address of touch2.

Phase3: In this phase, we need to store the cookie as a string, in a higher address than the buffer memory so that it is not overwritten by the randomized hexmatch function. We first input the byte code to set rdi to this string. Next, 4 bytes of zeroes to completely fill the buffer, followed by the address of rsp that holds the instruction so that it is returns there, rather than to test. Next we store the address of touch3, followed by the string in hex at the highest memory address, which is accessed by the set rdi instruction.

Phase4: For this phase, the stack is randomized and cannot execute code, so we need to use instructions in the code that already exist in order to pass rdi and pass touch2. So we use these instructions to move the cookie into rax than edi:

popq %rax

movq %rax ,%edi

ret

So we first fill the buffer with 40 bytes of zeros, then the address of pop rax instruction, then the cookie value which is popped into rax, then the move instruction address which moves rax into rdi, then the touch2 address.

Phase5: This uses even more roundabout instructions through a series of register moves. Here is the strategy:

1.Fill the buffer fully

2. Set rax=rsp, ret. Note current rsp address.

3.Set rdi=rax, ret (rdi now rsp address)

4.Pop rax, ret

5.0x48 popped into rax.

This is the offset of the cookie string address from the rsp address saved in rax. This will be placed in rsi.

6.Set edx=eax, ret

7. Set ecx=edx, ret

8.Set esi=ecx ==0x48 ,ret

9.call <add\_xy> which does: lea(%rdi,%rsi,1),%rax, ret //rax now holds address of cookie string

10.Set rdi=rax

11.Call touch 3 address

12.Store cookie as string, in address of rdi+rsi.