(1) program input that exploits the vulnerability, (2) brief explanation of why the exploit works, and (3) contents of the secret file.

Problem1
command=0xbf99abe4=3214519268
dataset=0xbf99abf0=3214519280
buffer=0xbf99abfc=3214519292
Command can overflow into dataset if there are more than 12 bytes
dataset can overflow into buffer if there are more than 12 bytes
(1) sssssssssssssss (2) You can create a segmentation fault by overloading the command array to access the dataset array with a 13 character input which since the space between command and dataset array is 12 bytes, 13 s will overflow from command array to dataset, and set dataset to “s” which is the key needed to access the secret message (3) “So it begins…”

Problem2
buffer= 0xbfd3e488 =3218334856
command=0xbfd3e48c=3218334860
Command can overflow into dataset if there are more than 4 bytes
(1) - 2147483447 (2) You can create an integer overflow by underflowing the command array to wrap around to 201 which since there is a constraint on an input being above 200, we use the minimum integer which is -2147483647 and add 200, which causes it to wrap around to 201 and access the file (3) “I have come here to chew bubblegum and hack programs, and I’m all out of bubblegum.”

Problem3

command=0xbfa0e4c4=3214992580
dataset=0xbfa0e4d0=3214992592
buffer=0xbfa0e4dc=3214992604
Command can overflow into dataset if there are more than 12 bytes
dataset can overflow into buffer if there are more than 12 bytes
(1) 385*s (2) You can create an buffer overflow by overflowing the temp buffer array with 384 s characters which will cause the strlen(temp_buffer) to cause an integer overflow because the max for an int8_t is 127 so it would then overflow 3 times so that length<size. Then the temp buffer will be copied into buffer of a smaller space which will cause an overflow into dataset then we add a singular s into dataset which will set dataset[0] to s and allow access to secret(3) “Oh, what a bug… what a lovely bug!”

Problem4
length=0xbfcea9a8=3217992104
secret=0xbfcea9a4=3217992100
input=0xbfcea99c=3217992092
length can overflow into secret if there are more than 4 bytes

Input can overflow into secret with more than 8 bytes
(1) %d %d %d %d %d %d %s (2) Because this program allows us to output the results of string formatting, using %d we can tell that input is 6 %d long so we can get to the secret value by reading 6 %d from the stack to get to the position of secret and use %s to read the string at that position (3) “Permission denied: cannot read ‘/home/student/secret4.txt’ … just kidding, this is actually the solution.”

Problem5
user.authorization_level=0xb771d918=3077691672
user.username=0xb771d91c=3077691676
minimum_level=0xbfe243a0=3219276704
user=0xbfe243a4=3219276708
buffer=0xbfe243ac=3219276716
user.authorization_level can overflow into user.username if there are more than 4 bytes
user.username can overflow into buffer if there are more than 141585040 bytes
minimum_level can overflow into user if there are more than 4 bytes
user can overflow into buffer if there are more than 12 bytes
(1) %201d %201d %201d %201d %n (2) Because this program allows us to output the results of string formatting, we can get to the variable user.authentication_level value by reading %201d 4 times for each byte from user.username to user.authentication_level which where 201 is the value we want to put in and %d reads bytes from the stack to get to the position of user.authentication_level and use %n to write the integer at that position (3) “Whew, you are finally done with the homework."