**CSCI476 - Homework1**

1. Suppose you could use all 94 practical symbols available on a regular keyboard.
   1. What is the total number of 8-character passwords that could be constructed? [1pt]
   2. Suppose an attacker could check one password every nanosecond, how long will he/she guess such a password on average? [1pt] Depends on method of guessing and the password selected. If the attacker was guessing passwords at random and guessed half of the passwords before finding the right one it would take

846.62 hours.

* 1. How could he/she speed up the guessing? [1pt] Use a password dictionary with real human made passwords in it.

1. A salesperson at a high-end computer security firm wants to sell you a protective cover for your passport, which contains an RFID tag inside storing your sensitive information. The salesperson's solution costs $79.99 and protects your passport from being read via radio waves while it is in your pocket. Explain how you can achieve the same thing for less than $79.99. [2pts]

Make a passport cover from duct tape and aluminum foil. Sandwich the aluminum foil in-between the two sticky sides of the duct tape and use that to make the case for much less. Make sure that the case completely encloses the passport.

1. Consider the following piece of C code:

int main(int argc, char \*argv[])

{

char continuee = 0;

char password[8];

strcpy(password, argv[1]);

if (strcmp(password, "CSCI476")==0)

continuee = 1;

if (continuee)

{

login();

}

}

void login()

{

char buf[100];

…

gets(buf);

…

}

* 1. Is this code vulnerable to a buffer overflow attack with reference to the variables *password[]* and *continuee*? If yes, describe how an attacker can achieve this? If not, please justify your answer. [2pts]

Yes. If an attacker inputted a string of length 9 or greater that contained only 1’s as the password then the ninth 1 would override the 0 in the memory address of continuee. This make the conditional check on continuee return true and allow the attacker to login.

* 1. To fix the problem, a solution is to remove the variable *continuee* and simply use the comparison for login. What kind of new buffer overflow attack can be achieved? You could use figures to illustrate your answers. [3pts]

Return Address Smashing. This can be done by writing malicious code into the char buff[100] that is created after login() is called. In order to write malicious code into the buffer you need to wait until prompted to do so by the gets(buf) call. Now all the attacker needs to do is rewrite a return address to point to the malicious code. Once the computer reaches this return address it will return to the point in memory where the malicious code starts and run the code.