CE4062/CZ4062 Computer Security

Tutorial 2 – Buffer Overflow

1. Answer the following questions
   a. What do vulnerability, exploit, and payload refer to?

   b. What could be the potential consequences of a buffer overflow attack?

   c. What are the steps to utilize a buffer overflow vulnerability to execute shellcode?

2. In the following program in Figure Q2, the function **get_plural** returns the plural of any input string. Assume the attacker can send arbitrary input to the main function. Please identify the possible buffer overflow vulnerabilities in this problem.

```
void get_plural(char* single, char* plural) {
  char* buf;
  plural[0] = '\0';
  int n = strlen(single);
  if (n == 0) return;
  buf = malloc(n+3);
  char last = single[n-1];
  if (last == 's')
    strcpy(buf, single);
  else if (last = 'h')
    sprintf(buf, "%ses", single);
  else
    sprintf(buf, "%ss", single);
  strcpy(plural, buf);
  free(buf)
}
void main(int argc, char* argv[]) {
  char plural_form[256];
  get_plural(argv[1], plural_form);
  printf("The plural of %s is %s\n", argv[1], plural_form);
}
```

Figure Q2

3. The following program in Figure Q3 is designed to generate a random number. It takes a password as input, but always fails to generate a random number. Luckily, this program is vulnerable to a buffer overflow attack. Our goal is to leverage this advantage to generate a random number. Please figure out a password that can achieve this.

4. A developer writes the following program in Figure Q4 for user authentication for his system. However, this program is vulnerable to buffer overflow attacks. Please give some examples of malicious input that an attacker can use to bypass the authentication.

```
char CheckPassword() {
  char good = 'N';
  char Password[100];
  gets(Password);
  return good;
}
int main(int argc, char* argv[]) {
  printf("Enter your password:");
  if(CheckPassword() == 'Y')
    printf("Your random number is %d\n", rand()%100);
  else{
    printf("You don't have the permission to get a random number");
    exit(-1);
  }
  return 0;
}
```

Figure Q3

```
int check_authentication(char *pwd) {
  int auth_flag = 0;
  char Password[] = "qwertyu";
  char buffer[8];
  strcpy(buffer, pwd);
  if (strncmp(buffer, Password, 8) == 0)
    auth_flag = 1;
  return auth_flag;
}
int main(int argc, char* argv[]) {
  if(check_authentication(argv[1]))
    printf("Access Granted\n");
  else{
    printf("Access Denied\n");
  }
  return 0;
}
```

Figure Q4