Application Software Security

Homework 01

Acknowledge any help and do not google for solution

Submission Instruction

Failure to follow the below instructions will result in a 20% penalty.

1. Create a folder named *"<First_name>_<Last_name>_HW_XX"* and put all your answers (e.g, source files) in this folder. For example: folder name *"David_Smith_HW_01"* for student name "David Smith" and assignment 01. No other object files or test files should be included.

2. Create a ZIP file of this folder with the same name (i.e., *"David_Smith_HW_01.zip"*) and submit it over Blackboard.

For question #2 - #5, screenshots are REQUIRED to demonstrate your answers.
Include your code & screenshots to your report and submit it to BB.

Question 01: Analyze the following code and indicate all potential vulnerabilities and suggest fixes for them.

[A]

```
int table[800];
int insert_in_table(int val, int pos){
    if(pos > sizeof(table) / sizeof(int)){
        return -1;
    }
   table[pos] = val;
    return 0;
}
```

[B]

```
int copy_something(char *buf, int len){
    char kbuf[800];
    if(len > sizeof(kbuf)){
        return -1;
    }
  return memcpy(kbuf, buf, len);
}
```

[C]

```
int myfunction(int *array, int len){
    int *myarray, i;

    myarray = malloc(len * sizeof(int));
    if(myarray == NULL){
        return -1;
    }

    for(i = 0; i < len; i++){
        myarray[i] = array[i];
    }
```

```
   return myarray;
}
```

[D]

```
int get_two_vars(int sock, char *out, int len){
   char buf1[512], buf2[512];
   unsigned int size1, size2;
   int size;

   if( recv(sock, buf1, sizeof(buf1), 0) < 0){
      return -1;
   }
   if(recv(sock, buf2, sizeof(buf2), 0) < 0){
      return -1;
   }

   /* packet begins with length information */
   memcpy(&size1, buf1, sizeof(int));
   memcpy(&size2, buf2, sizeof(int));

   size = size1 + size2;
   if(size > len){
      return -1;
   }

   memcpy(out, buf1, size1);
   memcpy(out + size1, buf2, size2);
   return size;
}
```

[E]

This example was taken from a security module for linux. This code runs in the kernel context:

```
int rsbac_acl_sys_group( enum  rsbac_acl_group_syscall_type_t call,
                         union rsbac_acl_group_syscall_arg_t arg)
{
  ...
 switch(call)
 {
     case ACLGS_get_group_members:
       if ((arg.get_group_members.maxnum<=0)|| !arg.get_group_members.group)
       {
          ...
          rsbac_uid_t * user_array;
          rsbac_time_t * ttl_array;

          user_array = vmalloc(sizeof(*user_array) *
          arg.get_group_members.maxnum);
          if (!user_array)
              return -RSBAC_ENOMEM;
          ttl_array = vmalloc(sizeof(*ttl_array) *
          arg.get_group_members.maxnum);
          if(!ttl_array)
          {
             vfree(user_array);
             return -RSBAC_ENOMEM;
          }

         err =
         rsbac_acl_get_group_members(arg.get_group_members.group,
                                     user_array,
```

```
                                                    ttl_array,
                                                    arg.get_group_members.max
                                                    num);
        ...
    }
```

Question 02: Answer the following questions

(a) Write a program that demonstrates the use of format strings for output functions. It should be a suitable example for an introductory programming course.

(b) Use the program in Format String Vulnerability lab and provide an input string that can modify the hex value of *test_val* to ***0xfeedbeef***.

Question 03: Create a web application that uses a MySQL database backend. Demonstrate it. Next, create a web application that uses a MySQL database backend that suffers from a SQL injection vulnerability. Demonstrate it. Correct the flaw in the previous problem by correctly sanitizing the input.

Question 04: Write a PHP script that suffers from a cross-site scripting vulnerability that can be exploited by a malicious link. Exploit the vulnerability to show a forged web page. Modify the previous script so that it retains the same functionality, but no longer suffers from a cross site scripting flaw.

Question 05: Write loadable kernel modules to set the followings firewall rules:

(a) Only block telnet traffic.

(b) Only block UDP packages on port > 2500

(c) Only allow web traffic.

(d) Only block web traffic from a certain domain, e.g., google.com, and allow all other traffic.