



SRM University-Delhi-NCR
Projects Committee
Computer Science and Engineering Department

APPROVAL FOR BINDING OF PROJECT REPORT
MINOR/MAJOR PROJECT (CS 4117)

B.Tech. Computer Science and Engineering _____

GROUP No : _____

SESSION 202__ / 202__

To be filled in by student

Student name(s) :

Signature : _____, _____

Registration Number :

Project title :

To be filled in by supervisor*

This is to certify that the project submitted and presented by the above mentioned student(s) is

☐ **COMPLETE & ACCEPTED**

☐ **INCOMPLETE**

☐ **NOT ACCEPTED**

and the draft of graduation project report has been corrected from all content flaws, typing errors and language mistakes.

Supervisor name :

Signature :

Date :

***NOTE:** The student may proceed with Ring binding ONLY after this section has been completed. The student shall submit three **copies** of Ring bound draft report for Minor/Major Project to the panel of examiners on the day of final presentation.

For Departmental Office use only

Date received :

Signatures :



**DEPARTMENT OF COMPUTER SCIENCE &
ENGINEERING**

**(MINOR PROJECT REPORT)
WEB APPLICATION ATTACKS**

COURSE CODE:	CS4117
COURSE TITLE:	MINOR PROJECT
SEMESTER & YEAR:	4th/ 7th
TEAM MEMBERS:	TANISHKA SAXENA SANYAM JINDAL
NAME OF SUPERVISOR:	DR. M. Mohan

CERTIFICATE

This is to certify that the minor project entitled “WEB APPLICATION ATTACKS” submitted under the guidance of Associate Professor Dr.M.Mohan, to the Department of Computer Science and Engineering of SRM University Delhi NCR, Sonipat, Haryana, India, by following students of 4th year 7th semester B.Tech CSE:

Tanishka Saxena Reg.No. 10320210054

Sanyam Jindal Reg.No. 10320210058

Dr. M. Mohan
Associate Professor
(Supervisor)

CANDIDATE DECLARATION

We, Tanishka Saxena and Sanyam Jindal hereby declare that the minor project report entitled "Web Application Attacks" submitted to SRM University, Sonipat is a record of original work carried out by me under the guidance of DR. M. Mohan. We assert that the contents of this report are based on our own research and analysis and have not been submitted for any other degree or examination at any other institution. We further declare that all external sources used in this project report have been duly acknowledged through proper citation and referencing.

Any contributions from collaborators or individuals have been appropriately recognized. We understand the importance of academic integrity and ethical conduct, and we affirm that the research conducted for this project has adhered to the ethical standards set forth by our institution.

We take full responsibility for any errors or omissions present in this report.

Tanishka Saxena

10320210054

Sanyam Jindal

10320210058

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to the Department of Computer Science and Engineering at SRM University Delhi NCR, Sonipat, Haryana, for providing us with the opportunity to undertake this project on Web Application Attacks.

We extend our heartfelt thanks to our project supervisor, Dr. M. Mohan for his invaluable guidance, encouragement, and continuous support throughout the duration of this project. Their expertise and insights have been instrumental in shaping our understanding and enhancing the quality of our work.

We also want to thank the faculty members of the Department of Computer Science and Engineering for their valuable feedback and suggestions during the course of our project.

Tanishka Saxena

Sanyam Jindal

ABSTRACT

This project delves into an extensive examination of web application attacks, focusing on in-depth analyses of various attack types such as SQL injection, phishing, DDoS, and sniffing. The primary objectives include understanding the underlying mechanisms of these attacks, exploring detection methodologies, and proposing effective prevention techniques. The research incorporates an exhaustive review of existing literature, encompassing research papers, articles, and case studies that contribute to the current understanding of web application security.

The project involves practical demonstrations of real-world attacks, illustrating the potential threats and vulnerabilities that web applications face. Detection mechanisms are a critical aspect of the study, and the project evaluates various models and techniques for identifying and mitigating web application attacks. The aim is to assess the effectiveness and limitations of each detection method in different scenarios.

Furthermore, the project explores preventive measures to enhance web application security. Prevention techniques range from secure coding practices and input validation to implementing robust authentication and access control.

LIST OF FIGURES

Figure 3.1: XAMPP control panel

Figure 3.2: Database for login credentials

Figure 3.3: Login Injection Successful

Figure 3.4: Database for URLs

Figure 3.5: URL validator

Figure 3.6: URL validator

Figure 3.7: URL validator

Figure 3.8: Finding open ports

Figure 3.9: Idle system load

Figure 3.10: Hping3

Figure 3.11: Load increment

Figure 3.12: Rendering of messages

Figure 3.13: HTML script tag

Figure 3.14: HTML script does not work

Figure 3.15: On error approach

Figure 3.16: Alert message

LIST OF TABLES

Table 2.1: Summary of Literature Review

LIST OF ABBREVIATIONS

SQL: Structured Query Language

DDoS: Distributed Denial of Service

DoS: Denial of Service

XSS: Cross-Site Scripting

HTTPS: Hypertext Transfer Protocol Secure

SSL: Secure Sockets Layer

TLS: Transport Layer Security

CSP: Content Security Policy

CSRF: Cross-Site Request Forgery

GDPR: General Data Protection Regulation

HIPAA: Health Insurance Portability and Accountability Act

PCI DSS: Payment Card Industry Data Security Standard

WC-PAD: Web Content Accessibility Guidelines - Portable and Accessible Documents

DOM: Document Object Model

DNS: Domain Name System

DHCP: Dynamic Host Configuration Protocol

ARP: Address Resolution Protocol

MAC: Media Access Control

IEEE: Institute of Electrical and Electronics Engineers

HTML: Hypertext Markup Language

VM: Virtual Machine

TABLE OF CONTENTS

TITLE	PAGE
CERTIFICATE	I
DECLARATION BY CANDIDATE	II
ACKNOWLEDGEMENT	III
ABSTRACT	IV
LIST OF FIGURES	V
LIST OF TABLES	VI
LIST OF ABBREVIATIONS	VII
CHAPTER 1: INTRODUCTION	1-4
CHAPTER 2: LITERATURE REVIEW	5-12
CHAPTER 3: REPORT ON THE PRESENT INVESTIGATION /	13-26
PROPOSED METHODOLOGY	27
CHAPTER 4: RESULTS/DISCUSSIONS	28-34
CHAPTER 5: CONCLUSION AND FUTURE WORK	35-36
REFERENCES	37

Chapter 1

INTRODUCTION

This project delves into the critical realm of web security, focusing on the analysis and implications of five prominent web attacks: phishing, Cross-Site Scripting (XSS), Distributed Denial of Service (DDoS), sniffing, and SQL injection. By exploring the characteristics, impacts, and preventive measures associated with each of these attacks, this study aims to enhance the overall understanding of web security vulnerabilities and to propose effective defence strategies for safeguarding online assets.

Web Application Attack and Security

A variety of malevolent actions aimed at taking advantage of holes in web applications are referred to as web application attacks. The security and integrity of web-based systems may be jeopardized by these attacks, which could result in illegal access, data leaks, and interruptions to services. Regular security assessments, code reviews, and the adoption of security protocols are essential components of a robust web application security strategy.

The policies and procedures used to safeguard web applications from cyberthreats, illegal access, and data breaches are together referred to as web application security. A key component of comprehensive cybersecurity is web application security, which focuses on defending websites, apps, and online-based systems from different attacks and vulnerabilities. Ensuring the availability, confidentiality, and integrity of the application and the data it processes is the aim.

There are several strong reasons why web application security is crucial.

- **Safeguarding sensitive information:**

Sensitive user data, including financial information, login passwords, and personal information, is handled by numerous web apps. It is essential to secure these data to stop any misuse and unauthorized access.

- **Preserving the privacy of users:**

Users anticipate careful handling of their personal data. Web application security protects user privacy and upholds trust by guaranteeing the confidentiality of user data.

- **Security of finances:**

Financial transactions are a common feature of web apps. Users and organizations alike may suffer financial damages as a result of security breaches. Ensuring the safety of payment details and transaction integrity is essential for everyone's financial health.

- **Company Image:**

Security issues, such website vandalism or data breaches, can seriously harm an organization's reputation. Maintaining credibility and confidence with users, clients, and partners is facilitated by a robust web application security posture.

- **Observance of Regulations:**

Strict laws covering the protection of user data are in place in many different sectors and geographical areas. Respecting various laws, including GDPR, HIPAA, or PCI DSS, is not only necessary to comply with the law but also necessary to stay out of trouble.

- **Preventing interruptions in service:**

Web applications can be made to malfunction by taking advantage of security flaws. assaults that cause service interruptions, such as denial-of-service (DoS) or distributed denial-of-service (DDoS) assaults, can have an impact on user experience and corporate operations.

In the realm of web application security, a multifaceted approach is essential to counteract diverse threats. Common threats, such as injection attacks (e.g., SQL injection), cross-site scripting (XSS), cross-site request forgery (CSRF), security misconfigurations, and authentication and session management issues, can expose vulnerabilities in the system. To mitigate these risks, various security measures are employed.

HTTPS (SSL/TLS) ensures secure communication, protecting the user's interaction with the web server. OAuth and OpenID Connect standards contribute to secure authorization and authentication processes, safeguarding user credentials and data. Security headers, such as Content Security Policy (CSP), provide an additional protective layer against specific types of attacks. the combination of comprehensive security measures, adherence to protocols, robust development practices, and effective incident response, coupled with regulatory compliance, creates a resilient defense against the dynamic landscape of web application threats.

Common Threats

- **Phishing**

Web Crawling based Phishing Attack Detection

Phishing is a form of cyber-attack where fraudsters use fake emails, messages, or websites to deceive people into revealing personal information, like passwords or credit card numbers. They often pose as trusted entities to gain victims' trust and exploit human psychology to trick them into giving up sensitive data or downloading harmful software. This type of scam is widespread and underscores the importance of robust security measures and user vigilance in the digital age.

The research paper introduces WC-PAD, a novel Web Crawler based Phishing Attack Detector, addressing the limitations of existing anti-phishing measures. Emphasizing the need for effective software-based solutions, the paper highlights WC-PAD's ability to achieve 98.9% accuracy in detecting standard phishing attempts and zero-day attacks. By leveraging web traffic, content, and URL features, WC-PAD effectively distinguishes between phishing and non-phishing websites.

- **XSS**

A Comparative Analysis of Cross Site Scripting (XSS) Detecting and Defensive Techniques

XSS, or Cross-Site Scripting, is a web security vulnerability where attackers inject malicious scripts into web pages, enabling them to hijack user sessions, steal data, or redirect users to harmful sites. It often exploits unvalidated user inputs and can be prevented through strict input validation and output encoding practices.

This study discusses the XSS attack, its taxonomy, and its incidence. In addition, the paper presents the XSS mechanisms to combat vulnerabilities, calling for continued research to address Stored XSS and enhance defense against DOM-based XSS, possibly leveraging data mining and machine learning for advanced detection capabilities.

- **SQL Injection**

Research on SQL Injection Attack and Prevention Technology Based on Web

SQL Injection is a type of cyber-attack that involves inserting malicious SQL code into an application's input fields, exploiting vulnerabilities in the application's database layer. This attack allows unauthorized access to sensitive data, manipulation of database content, and potentially the complete takeover of the targeted system.

This research paper examines the widespread risk of SQL injection attacks in the context of database security, particularly within B/S mode application development. Highlighting the oversight of user input validation among programmers, it emphasizes the importance of robust security protection technologies. Through a comprehensive analysis of attack principles, forms, and preventive strategies, the paper offers practical insights supported by real-world examples.

- **DDoS**

DDoS Detection and Prevention Based on Artificial Intelligence Techniques

DDoS, or Distributed Denial of Service, is a type of cyber-attack that aims to overwhelm a target system or network with a flood of internet traffic, rendering it inaccessible to legitimate users. This is typically achieved by utilizing a network of compromised devices, known as a botnet, to generate a massive volume of traffic directed at the target, thereby disrupting its normal functioning. DDoS attacks can lead to severe downtime, financial losses, and reputational damage for the targeted organization or individual. Preventative measures and robust security protocols are crucial in mitigating the impact of DDoS attacks.

In the paper, we survey on the latest progress on the DDoS attack detection using artificial intelligence techniques and give recommendations on artificial intelligence techniques to be used in DDoS attack detection and prevention.

- **Sniffing**

A survey on sniffing attacks on computer networks

Sniffing refers to the act of intercepting and monitoring data packets transmitted over a network. Typically employed by attackers or security professionals, sniffing allows the capture of sensitive information, such as usernames, passwords, or other confidential data, leading to potential security breaches. Sniffing can be conducted through various methods, including the use of specialized software or hardware, and is often carried out to gain unauthorized access to network traffic for malicious or surveillance purposes. Preventing and detecting sniffing activities is crucial to ensuring the security and privacy of network communications.

This research paper aims to provide a comprehensive comparison of various sniffing attacks

prevalent in different networking levels, such as Media Access Control (MAC) flooding, Dynamic Host Configuration Protocol (DHCP) attacks, Address Resolution Protocol (ARP) spoofing, and Domain Name Server (DNS) poisoning. The study focuses on assessing the effectiveness of recovery measures against each attack, contributing to the understanding of network security and aiding in the development of robust defence strategies

Chapter 2

LITERATURE REVIEW

Phishing

Web Crawling based Phishing Attack Detection

Literature Review:

- Phishing is a criminal offense that violates the rule of Confidentiality, Integrity, and Availability, and it poses a threat to individuals and organizations.
- Hardware-based approaches for phishing detection are accurate but expensive, leading to a preference for software-based approaches.
- Blacklist and white list methods are commonly used for phishing detection, but they require manual list maintenance, leading to the adoption of automatic detection approaches like machine learning and heuristics.
- Web structure and web content-based methods, as well as heuristic approaches, are used for identifying phishing URLs.
- Web crawlers, typically used for information extraction, are a new approach in phishing research and can address zero-day phishing attacks.
- Existing approaches, such as plug-ins for browsers and Netcraft phishing detection, have limitations in detecting zero-day phishing attacks.
- WC-PAD is a proposed three-phase phishing attack detection approach that utilizes DNS blacklist, web crawlers, and heuristics for precise detection of phishing websites.
- Experimental analysis of WC-PAD shows a high accuracy of 98.9% in detecting both phishing and zero-day phishing attacks.
- WC-PAD incorporates features like web traffic analysis, URL analysis, and web content analysis for classification of websites.

Areas of Improvement

- The paper does not provide details about the size or diversity of the datasets used for experimental analysis.
- The paper does not discuss the computational resources required for implementing the WC-PAD approach.
- The paper does not mention the false positive or false negative rates of the WC-PAD approach.
- The paper does not provide information about the scalability of the WC-PAD approach for

handling large-scale web crawling and detection.

- The paper does not discuss the potential challenges or limitations of using web crawlers for feature extraction and phishing attack detection.
- The paper does not compare the performance of the WC-PAD approach with other existing phishing detection approaches.
- The paper does not address the potential limitations or vulnerabilities of the WC-PAD approach in detecting sophisticated or evolving phishing attacks.

XSS

A Comparative Analysis of Cross Site Scripting (XSS) Detecting and Defensive Techniques

Literature Review

- The paper provides a comparative analysis of Cross Site Scripting (XSS) detecting and defensive techniques.
- It discusses the XSS attack, its taxonomy, and its incidence.
- The paper presents various XSS mechanisms used to detect and prevent XSS attacks.
- It mentions the future work of developing a defensive mechanism using data mining and machine learning techniques to detect and prevent Stored XSS and DOM-based XSS attacks.
- Gupta and Gupta proposed an automated detection system for discovering malicious JavaScript code injection vulnerabilities in web applications.
- Huajie Xu et al. proposed a defensive model based on user browsing behavior and website logic structure to protect against XSS attacks.
- Parameshwaran et al. developed the DEXTERJS tool to detect and prevent DOM-based XSS vulnerabilities.
- The paper includes tables that show the strengths and weaknesses of existing XSS detection and prevention techniques, as well as a comparison of these techniques in terms of exploitation location, attack types, and XSS taxonomy.

Areas of Improvement

- The techniques discussed in the paper do not detect or prevent DOM-based XSS attacks.
- The analysis is focused only on server-side vulnerabilities and does not consider other untrusted contexts.
- The techniques are targeted only towards Java-based web applications, limiting their applicability to other programming languages.
- Modifying the source code of the program is required, which may be ineffective when the source code is not available.
- The techniques do not track information flow across the web page, potentially missing vulnerabilities.

SQL Injection

Research on SQL Injection Attack and Prevention Technology Based on Web

Literature Review

- SQL injection attacks are a common security vulnerability in web applications that allow attackers to execute malicious SQL commands by inserting them into user input fields.
- There are two main forms of SQL injection attacks: direct injection and indirect injection. Direct injection involves inserting code directly into user input variables concatenated with SQL commands, while indirect injection involves injecting malicious code into strings stored in tables or as original documents.
- SQL injection attacks can lead to unauthorized access to databases, data tampering, and even data destruction. It is crucial for web developers to understand the principles of SQL injection and implement proper security measures to protect their databases.
- Preventive measures against SQL injection attacks include properly writing SQL statements, filtering special characters, and using parameterized queries or prepared statements.

Areas of Improvement

- The paper does not provide a comprehensive analysis of the different types of SQL injection attacks and their specific vulnerabilities.
- The paper does not discuss the potential impact and consequences of SQL injection attacks on different types of databases and applications.
- The paper does not provide detailed guidelines or best practices for preventing SQL injection attacks.
- The paper does not discuss the limitations or challenges of implementing the suggested prevention methods.
- The paper does not provide empirical evidence or case studies to support the effectiveness of the suggested prevention methods.

DDoS

DDoS Detection and Prevention Based on Artificial Intelligence Techniques

The paper provides a comprehensive analysis of the threat posed by Distributed Denial of Service (DDoS) attacks to the Internet, emphasizing the challenges associated with detection and prevention. It delves into the complexities of various attack methodologies and the evolving nature of DDoS attacks, considering the implications of emerging technologies such as cloud computing, IoT, and artificial intelligence.

Literature Review

1. Thorough examination of the various types of DDoS attacks, including IP spoofing, flooding attacks, and DNS amplification attacks.
2. Comprehensive analysis of the evolving trends in DDoS attacks, highlighting the increasing scale and sophistication of these attacks.
3. In-depth exploration of artificial intelligence techniques, particularly machine learning algorithms, for the detection and classification of DDoS attacks.
4. Classification of various DDoS attack types and the impact of each on network resources and system availability.
5. Identification of specific AI techniques, including machine learning algorithms, neural networks, and deep learning, for effective DDoS attack detection.
6. Insightful discussion on the integration of AI with technologies like Hadoop for efficient processing and analysis of large-scale DDoS attack data.

Areas of Improvement

1. Limited discussion on specific case studies or real-world applications of the proposed detection and prevention methods.
2. More detailed information on the challenges and limitations of implementing AI-based solutions for DDoS attack detection would have been beneficial.

SNIFFING

A survey on sniffing attacks on computer networks

This research paper provides a comprehensive analysis of packet sniffing and various related attacks, including MAC flooding, DHCP attacks, ARP spoofing, DNS poisoning, and their respective defence mechanisms. The paper systematically outlines the types of attacks, the vulnerabilities they exploit, and the potential impact on network security. It also discusses the tools commonly employed by attackers and proposes several countermeasures and detection techniques to mitigate the risks associated with packet sniffing. Here is a detailed review of the research paper:

Literature Review

1. The paper offers a comprehensive overview of various network vulnerabilities and the potential risks associated with different types of packets sniffing attacks.
2. The content is organized in a structured manner, allowing readers to follow the progression of ideas and concepts easily. Each section is clearly defined, making the information accessible to readers with varying levels of expertise in the field.
3. The paper provides an in-depth analysis of each type of attack, including detailed explanations of their mechanisms, potential impacts, and recommended defence strategies. This thorough analysis enhances the reader's understanding of the complexities involved in network security and intrusion detection.
4. The inclusion of practical examples and real-world scenarios, along with the suggested solutions and detection techniques, enhances the practical applicability of the research.

Areas for Improvement

1. Current Research References: Including recent research findings or the latest developments in the field would strengthen the paper's relevance and demonstrate a comprehensive understanding of the current state of network security.
2. Real-world Case Studies: Integrating real-world case studies or practical examples of successful implementation of the proposed solutions would provide readers with a clearer perspective on the practical implications of the research.

SUMMARY OF LITERATURE REVIEW

S. No.	Title Of Paper	Journal / Conference / Book Chapter	Remarks
1.	Research on SQL Injection Attack and Prevention Technology Based on Web	2019 International Conference on Computer Network, Electronic and Automation (ICCNEA)	<ul style="list-style-type: none"> There are two main forms of SQL injection attacks: direct injection and indirect injection. SQL injection attacks can lead to unauthorized access to databases, data tampering, and even data destruction. Preventive measures against SQL injection attacks include properly writing SQL statements, filtering special characters, and using parameterized queries or prepared statements.
2.	WC-PAD: Web Crawling based Phishing Attack Detection	2019 International Carnahan Conference on Security Technology (ICCST)	<ul style="list-style-type: none"> WC-PAD is a proposed three-phase phishing attack detection approach that utilizes DNS blacklist, web crawlers, and heuristics for precise detection of phishing websites. WC-PAD incorporates features like web traffic analysis, URL analysis, and web content analysis for classification of websites.
3.	A survey on sniffing attacks on computer networks	2017 International Conference on Intelligent Computing and Control (I2C2)	<ul style="list-style-type: none"> Provides a comprehensive analysis of packet sniffing and various related attacks, including MAC flooding, DHCP attacks, ARP spoofing, DNS poisoning Provides an in-depth analysis of each type of attack, including detailed explanations of their mechanisms, potential impacts, and recommended defence strategies.

Table: 2.1

S. No.	Title Of Paper	Journal / Conference / Book Chapter	Remarks
4.	A Comparative Analysis of Cross Site Scripting (XSS) Detecting and Defensive Techniques	The 8th IEEE International Conference on Intelligent Computing and Information Systems (ICICIS 2017)	<ul style="list-style-type: none"> • The paper provides a comparative analysis of Cross Site Scripting (XSS) detecting and defensive techniques. • The techniques are targeted only towards Java-based web applications, limiting their applicability to other programming languages. • The techniques do not track information flow across the web page, potentially missing vulnerabilities.
5.	DDoS Detection and Prevention Based on Artificial Intelligence Techniques	2017 3rd IEEE International Conference on Computer and Communications	<ul style="list-style-type: none"> • Provides a comprehensive analysis of the threat posed DDoS attacks to the Internet, emphasizing the challenges associated with detection and prevention. • Comprehensive analysis of the evolving trends in DDoS attacks. • In-depth exploration of artificial intelligence techniques for the detection and classification of DDoS attacks.

Table: 2.1

Chapter 3

REPORT ON THE PRESENT INVESTIGATION / PROPOSED METHODOLOGY

SQL Injection

SQL injection typically occurs when user inputs, such as those from login forms, are directly incorporated into SQL queries without proper validation or sanitization.

An attacker may input specially crafted strings that include SQL code. For example, instead of entering a valid username and password, an attacker might input:

' OR '1'='1'; --

If the injected SQL code is successful, the query could return results even if the entered username and password are incorrect. This could lead to unauthorized access, data theft, or manipulation.

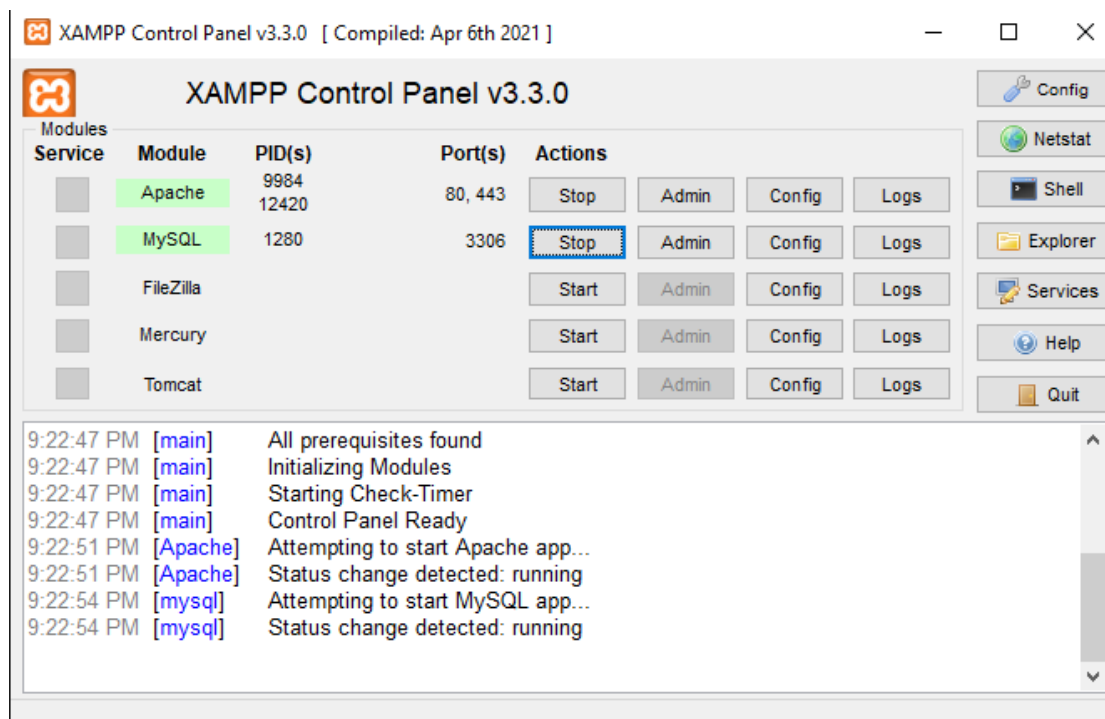


Figure 3.1: XAMPP control panel

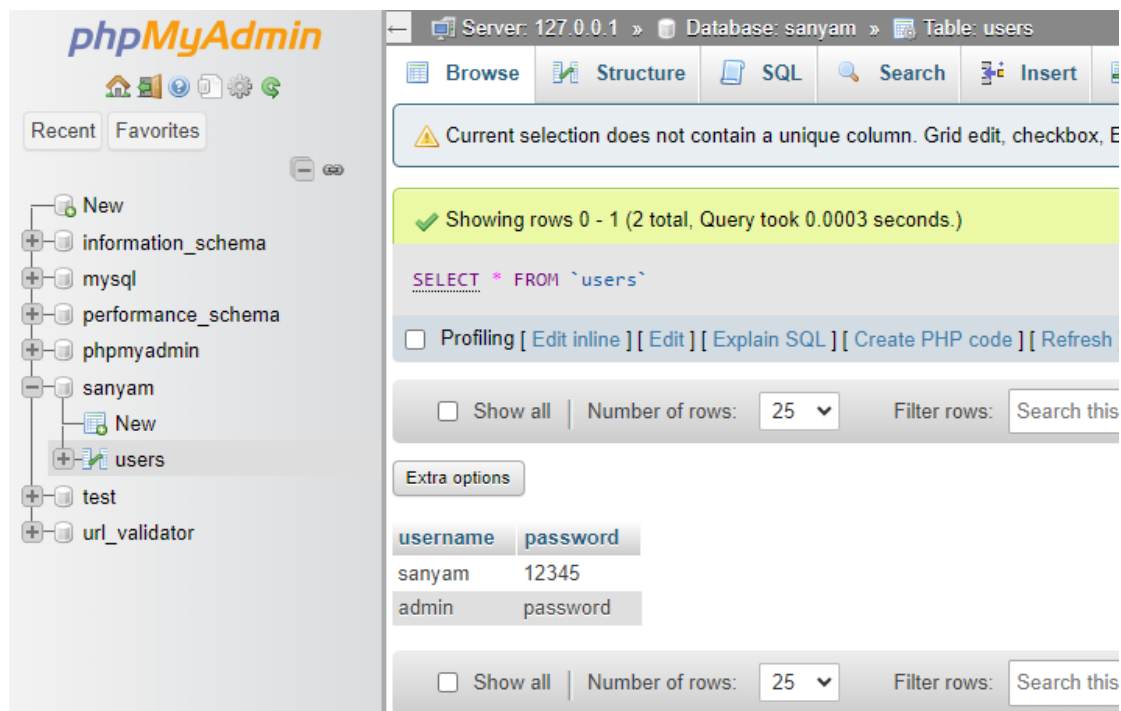


Figure 3.2: Database for login credentials

> Create a login page using HTML, CSS AND PHP
 > Using PHPmyadmin create a database named “sanyam” located at <http://localhost/phpmyadmin/>
 > Using Xampp control panel establish connection between the page and phpMyAdmin
 > Access the login page at <http://localhost/loginpage/index.html>
 and try logging in using correct credentials, this results into successful login.

> Now, if we try to perform sql injection using command “ ' OR '1'='1' -- ” on the unprotected page the attack gets performed because In a normal statement, SQL queries are constructed by embedding user inputs directly into the query string. This means that the query is assembled dynamically and may include user-provided data without proper validation or sanitization like (OR 1=1) which is always TRUE.

```
$sql = "SELECT * FROM users WHERE username='$username' AND password='$password'";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    echo "Login successful!";
} else {
    echo "Login failed.";
}
}
$conn->close();
?>
```

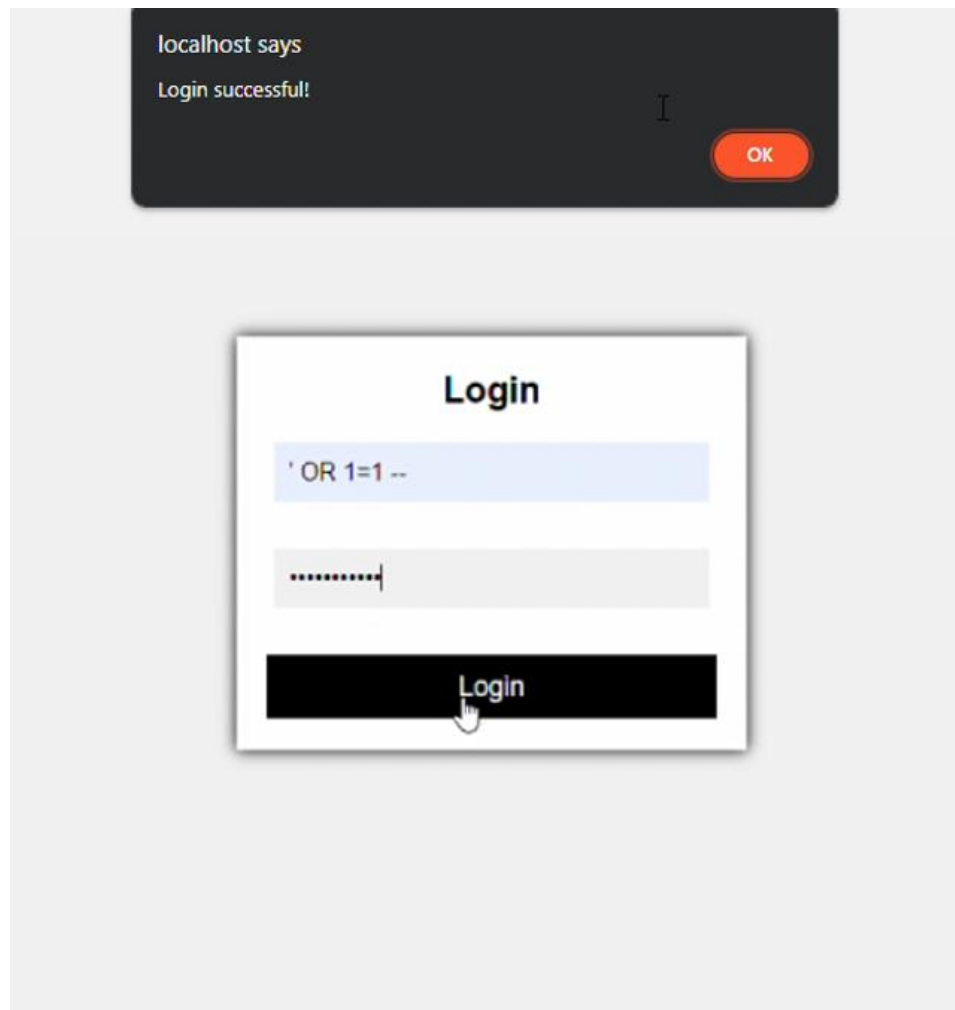



Figure 3.3: Login Injection Successful

Phishing

Phishing URLs are a critical component of phishing attacks, which are fraudulent attempts to obtain sensitive information, such as usernames, passwords, and financial details, by posing as a trustworthy entity.

A URL validator checks whether a given string conforms to the structure and syntax of a valid URL. It verifies whether the provided URL is well-formed and adheres to the standards defined by the Uniform Resource Identifier (URI) specification.

URL validation is a crucial step in preventing potential security vulnerabilities, such as URL injection attacks. Validating URLs helps mitigate the risk of malicious users inserting harmful or unintended URLs into systems.

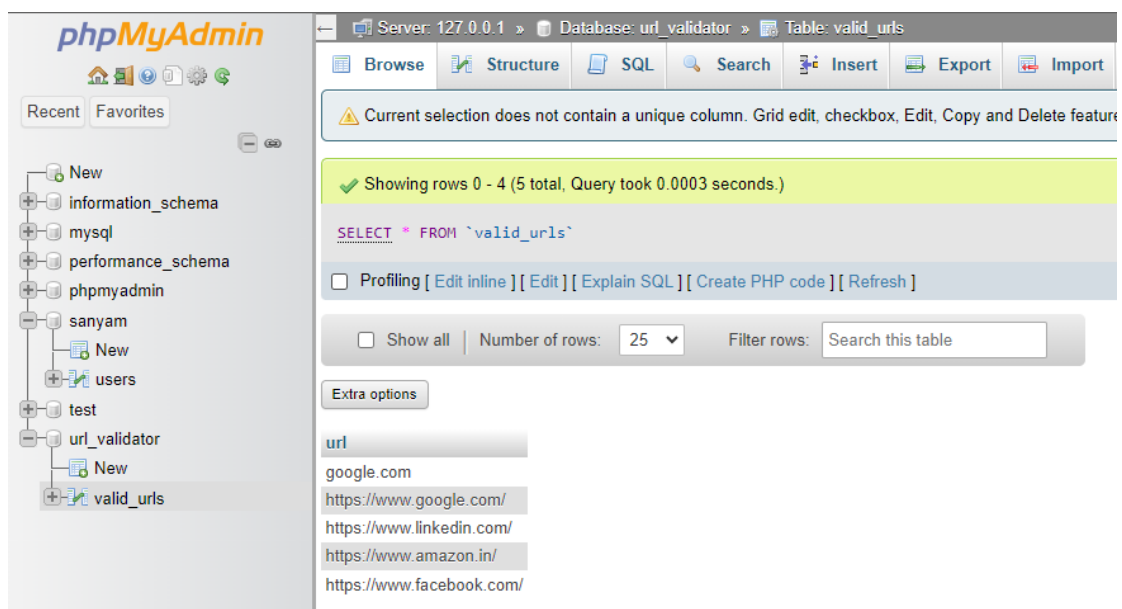


Figure 3.4: Database for URLs

- > Create a URL validator using HTML, CSS AND PHP
- > Using PHPmyadmin create a database named “url_validator” located at <http://localhost/phpmyadmin/>
- > Using Xampp control panel establish connection between the page and phpMyAdmin
- > Access the page at http://localhost/url_comparison/index.html

>Upon entering an URL like <https://www.facebook.com/>
The output comes out as “valid URL”

>Upon trying URL like <https://www.facebook.fb.com/>
The output “Invalid/phishing url will be received because the datatbase does not have this URL present.”

```

<?php
// Database connection parameters
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "url_validator";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $input_url = $_POST["url"];

    // Query to check if the entered URL is in the database
    $sql = "SELECT * FROM valid_urls WHERE url = '$input_url'";
    $result = $conn->query($sql);

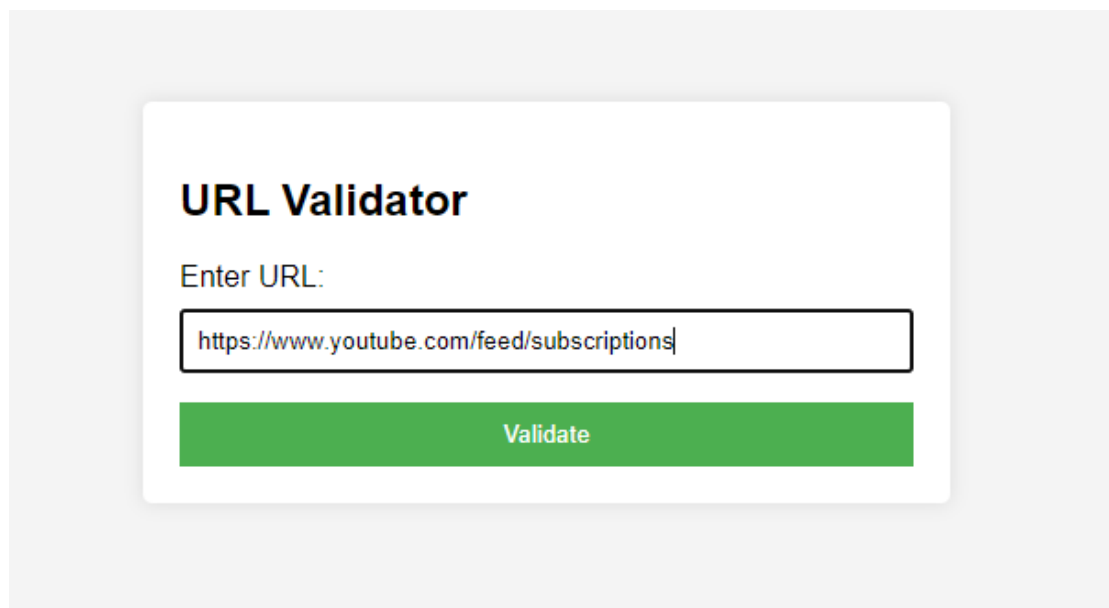
    if ($result->num_rows > 0) {
        // echo "Login successful!";
        echo '<script>alert("Valid URL")</script>';
    } else {
        // echo "Login failed.";
        echo '<script>alert("Invalid / Phishy URL")</script>';
    }
}

// Close the database connection
$conn->close();
?>

```

We can follow a different approach as well where we are comparing the URL on the basis of valid patterns to predict any invalid URL like:

- Check for common phishing patterns in the URL
- Check if the URL uses IP address instead of domain name
- Check for unusual characters in the domain name
- Check if the URL uses an IP address in the path
- Additional checks based on specific criteria, such as subdomains, length, etc.

A screenshot of a web application titled "URL Validator". It features a text input field containing the URL "https://www.youtube.com/feed/subscriptions" and a green "Validate" button below it. The interface is clean and modern, with a white card on a light gray background.

URL Validator

Enter URL:

Validate

Figure 3.5: URL validator

Sniffing

Network sniffing or packet sniffing, involve the interception and analysis of data packets as they travel over a network. Attackers use network sniffing to capture sensitive information, such as usernames, passwords, or other confidential data. The process involves monitoring and inspecting the data packets transmitted between devices on a network.

The attacker uses packet capture tools, such as Wireshark, tcpdump, or ettercap, to capture and analyze the data packets traversing the network.

Sniffing attacks are more effective on unencrypted traffic. If sensitive information is transmitted in plain text, the attacker can easily view and extract the data from the captured packets.

In some cases, attackers may perform Man-in-the-Middle attacks, positioning themselves between the communication flow of two parties. This allows them to intercept and manipulate the communication.

Signup new user

Please do not enter real information here.

If you press the submit button you will be transferred to a secured connection.

Username:	<input type="text" value="tester"/>
Password:	<input type="password" value="...."/>
Retype password:	<input type="password" value="...."/>
Name:	<input type="text" value="Testcase"/>
Credit card number:	<input type="text" value="123-456-789"/>
E-Mail:	<input type="text" value="testcase@gmail.com"/>
Phone number:	<input type="text" value="9879879879"/>
Address:	<input type="text" value="Test location address."/>
<input type="button" value="signup"/>	

Figure3.6: Sign Up page

If you are already registered please enter your login information below:

Username :	<input type="text" value="test"/>
Password :	<input type="password" value="...."/>
<input type="button" value="login"/>	

Figure: Login Page

- > Creating a vulnerable page which uses “HTTP protocol” instead of the secured one “HTTPS”.
- > Using login details to login into the webpage
- > Wireshark displays a real-time list of captured packets, including details such as source and destination IP addresses, protocols used, and data payload.
- > Using “Wireshark” we can easily analyse the HTTP stream to find out the saved username and password.

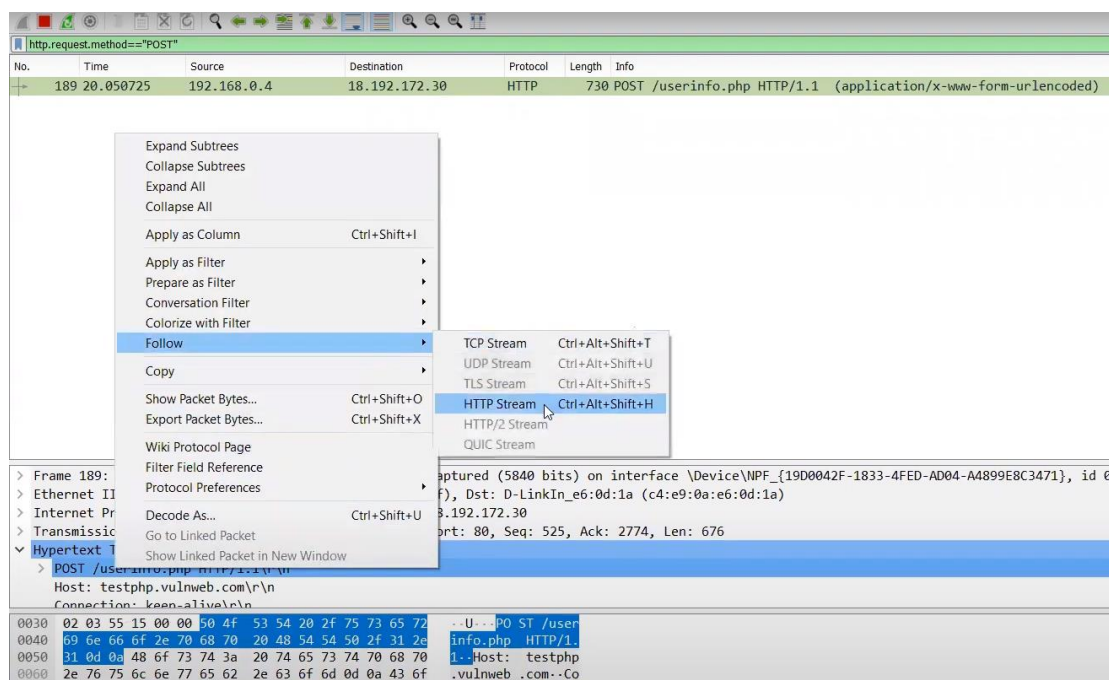


Figure 3.7: Wireshark Interface

Distributed Denial of Service

A Distributed Denial of Service (DDoS) attack is a malicious attempt to disrupt the regular functioning of a targeted server, service, or network by overwhelming it with a flood of internet traffic. Unlike a traditional Denial of Service (DoS) attack, where a single source is used to flood the target, DDoS attacks involve multiple sources, often distributed across various locations or devices.

Setup of Virtual Environment: Virtualization Platform:

Programme: Oracle VirtualBox; Function: Provides a virtualization platform for the creation and administration of virtual machines.

Virtualized systems (VMs):

Linux virtual machine (VM):

Operating system: A Linux distribution (Kali Linux).

Goal: Acted as the attacker machine in your demonstration

Tools: Outfitted with the required equipment (e.g., hping3, nmap) for reconnaissance and DDoS attack launch.

Virtual Machine for Windows:

Operating System: Windows distribution (Windows 10).

Goal: Acted as the target machine in your demonstration.

Vulnerabilities: Open ports or services in order to highlight possible weaknesses.

Network Configuration:

The two virtual machines and the host computer were connected to the network via Host-Only Adapter mode.

Security Procedures:

Isolation: To limit the impact of the DDoS demonstration, it was made sure the virtual machines were separated from the host system and other networks.

Snapshots: In order to enable a speedy return to a known condition, it was thought to take snapshots prior to the demonstration.

Demonstration Flow:

Ping Test: To verify the network connection, ping was used to check the connectivity between the Windows and Linux virtual machines.

Nmap Scan: On the Windows virtual machine, Nmap was used to find open ports and possibly vulnerable services.

Tool Used: Employed hping3 on the Linux VM to simulate a DDoS attack against the Windows VM.

```

root@tani: ~
File Actions Edit View Help
hping3 --scan 1-65535 192.168.56.103 -S --rand-source
Scanning 192.168.56.103 (192.168.56.103), port 1-65535
65535 ports to scan, use -V to see all the replies
+-----+-----+-----+-----+
|port| serv name | flags | ttl | id | win | len |
+-----+-----+-----+-----+
All replies received. Done.
Not responding ports:

root@tani: ~
nmap 192.168.56.101
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-07 21:46 IST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.56.101
Host is up (0.00030s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
MAC Address: 08:00:27:48:48:82 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 1.70 seconds

```

Figure 3.8: Finding open ports

Commands to begin the attack:

hping3 --scan 1-65535 192.168.56.103 -S --rand-source
used to find open ports.

The `--rand-source` is used to scan the attacked machine without revealing the true IP address of the attacker. The command runs and returns the open ports as shown in the fig. On the other hand the Windows VM's performance monitor is stable and low, representing that no attack has yet taken place.

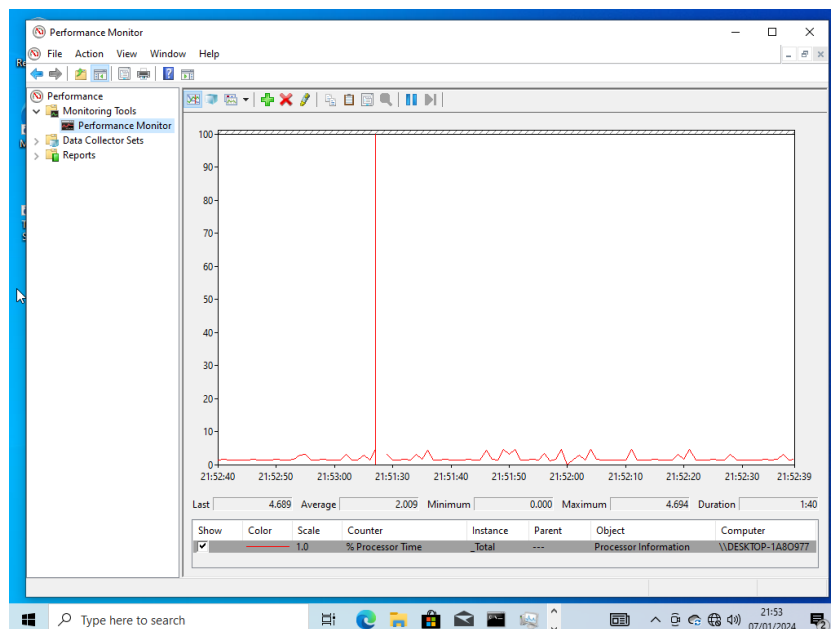


Figure 3.9: Idle system load

To begin the DDoS attack, we run a hping3 command to flood the victim machine:


```

root@tani: ~
File Actions Edit View Help
49664 : .S..A... 128 45084 65392 46
49665 : .S..A... 128 45340 65392 46
49666 : .S..A... 128 45596 65392 46
49667 : .S..A... 128 45852 65392 46
49668 : .S..A... 128 46108 65392 46
49669 : .S..A... 128 46364 65392 46
49670 : .S..A... 128 46620 65392 46
All replies received. Done.
Not responding ports:

root@tani: ~
# hping3 -S 192.168.56.102 -a 192.168.56.101 -p 135 --flood
HPING 192.168.56.102 (eth0 192.168.56.102): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
stop

^C
-- 192.168.56.102 hping statistic --
2028518 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

root@tani: ~
# hping3 -c 10000 -d 10000 -S -p 135 --flood --rand-source 192.168.56.101
HPING 192.168.56.101 (eth0 192.168.56.101): S set, 40 headers + 10000 data by
tes
hping in flood mode, no replies will be shown

```

Figure 3.10: Hping3

As soon as the command is run the victim's machine has started to flood with packets and the performance monitor shows a lot of activities on the system inspite that the system itself is idle.

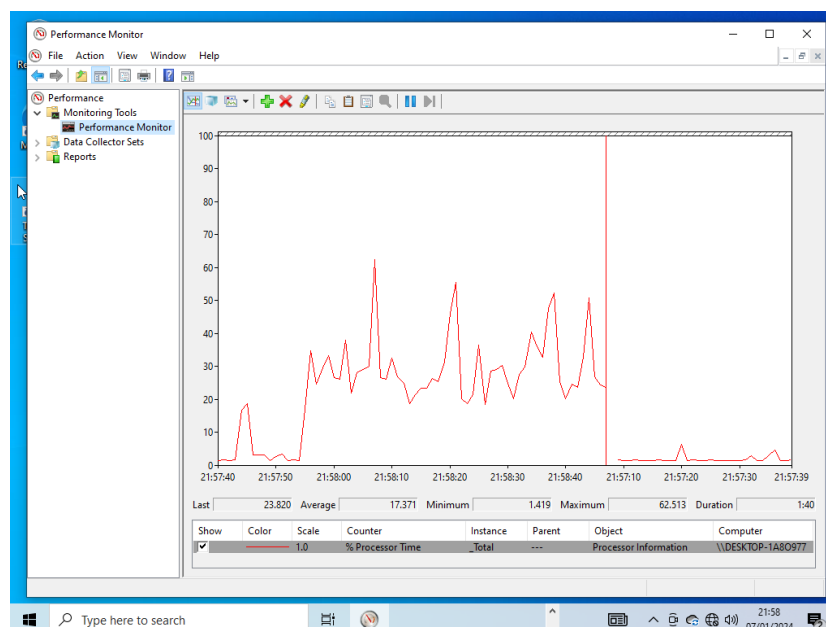


Figure 3.11: Load increment

On the Linux VM, now we run Wireshark.

Wireshark is a powerful open-source network protocol analyzer that allows users to capture and inspect the data traveling back and forth on a network in real-time. In the context of your DDoS attack demonstration between a Linux VM (attacker) and a Windows VM (target), Wireshark can be a valuable tool for capturing and analyzing network traffic. In the below figure it shows how the attackers IP address changes after each interval, resulting in a distributed DoS attack

Cross Site Scripting

Using a simple application example, we aim to explain that XSS attacks inject and execute malicious JavaScript code on users' devices.

In the demonstration we see a simple application in which we can send messages with some text and an image and we can have multiple messages here which are then simply rendered below each other. This can be imagined as messages and image URLs are sent to a server and then there they are stored in a database just how a vast majority of websites work.

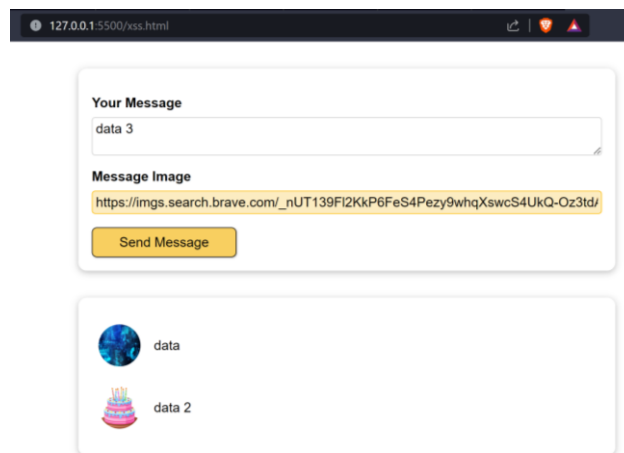


Figure 3.12: Rendering of messages

A cross site scripting app is about executing JavaScript code on our user's device and a trivial attack pattern is adding script tags here in the message and you then have some malicious code here that is inserted between the script tags.

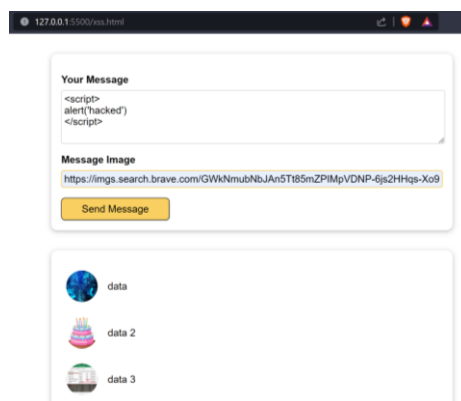


Figure 3.13: HTML script tag

The messages sent by the user are validated

```
if (
  !userMessage ||
  !imageUrl ||
  userMessage.trim().length === 0 ||
  imageUrl.trim().length === 0
) {
```

```

    alert('Please insert a valid message and image.');
```

```

    return;
```

```

  }
```

and then added to a user message array (in reality they would then probably be sent to a server and stored in a database and be fetched when the website is loaded).

```

userMessages.push({
  text: userMessage,
  image: imageUrl,
});
```

Whenever we have messages we render them at some point and here rendering the messages simply means looping through all the loaded messages and build a long string full of list items which is then added in inner HTML.

```

function renderMessages() {
  let messageItems = "";
  for (const message of userMessages) {
    messageItems = `
    ${messageItems}
    <li class="message-item">
      <div class="message-image">
        
      </div>
      <p>${message.text}</p>
    </li>
    `;
  }
}
```

messages are rendered here within a HTML in the end and therefore in a HTML of course interprets everything here as HTML so adding a HTML script tag might work.

But, it fails as modern day web pages are now aware of this pattern. Therefore, even though the code injected by the live server is shown, it does not work.

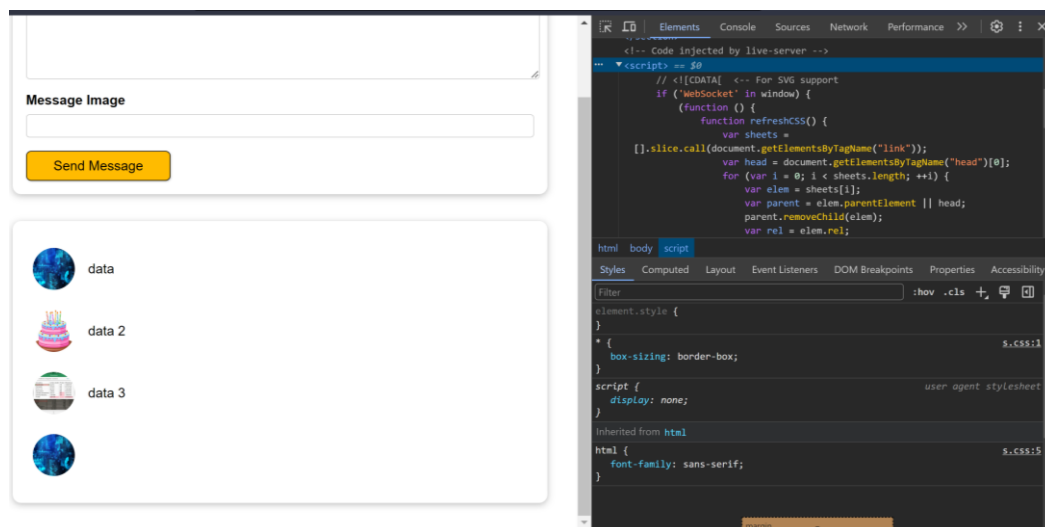


Figure 3.14: HTML script does not work

Trying another approach- We start with an invalid image URL. So this URL doesn't exist but I add double quotes here, effectively closing off the source attribute. Now we add a new attribute which is officially supported and that's the on error attribute. On an error wants JavaScript code which executes whenever loading the image fails. So, by using a invalid URL here, I force this loading process to fail and then here I could add my JavaScript code without script text. As you see this hacked alert .

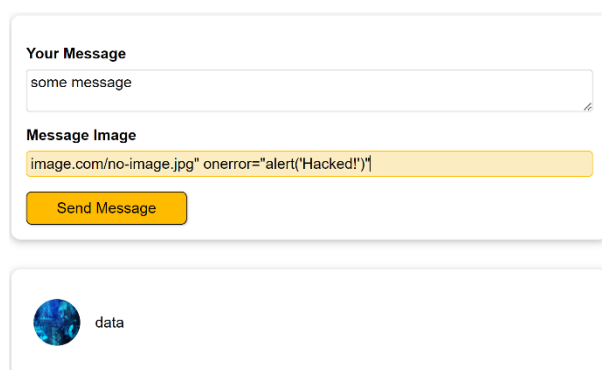


Figure 3.15: On error approach

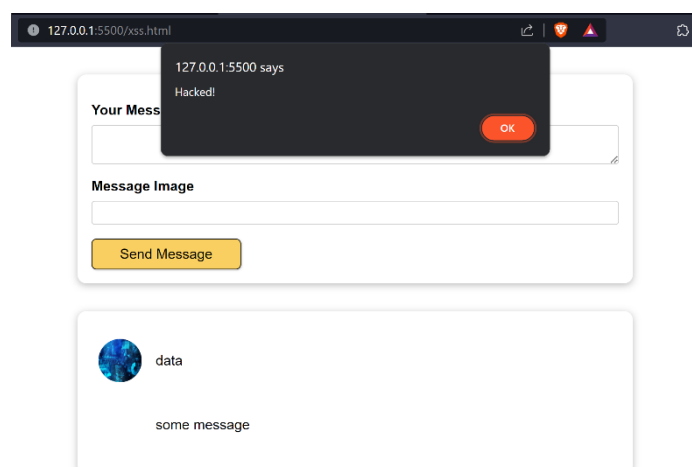
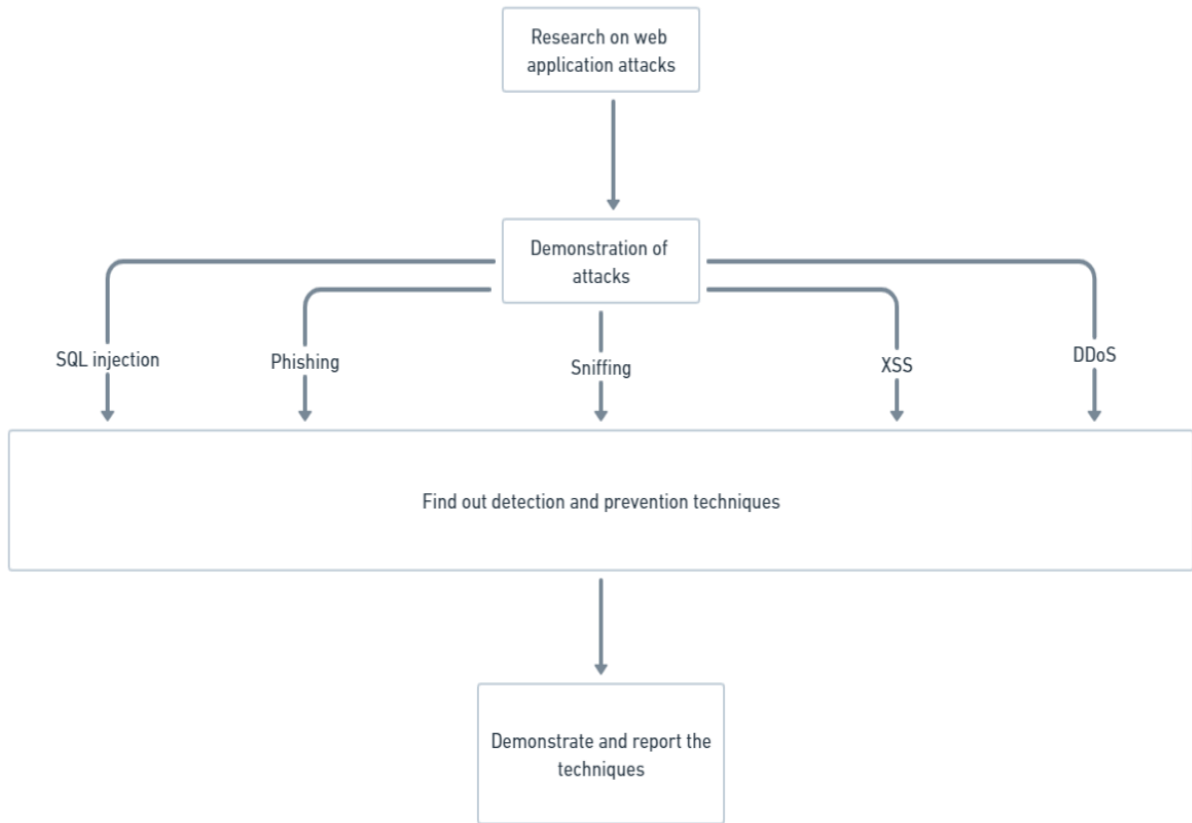


Figure 3.16: Alert message

It is recommended that developers should avoid setting image sources as strings and instead opt for selecting images after they are rendered. It is also advised that sanitizing user input using packages like Node.js's "node-sanitize" to check for malicious code patterns and remove them before the data reaches the client-side.

There are potential security risks associated with third-party libraries which highlights the importance of using trusted ones and being aware of vulnerabilities. The audience is encouraged to minimize their reliance on third-party libraries and, as a positive side effect, reduce the number of libraries to ship and create faster websites.

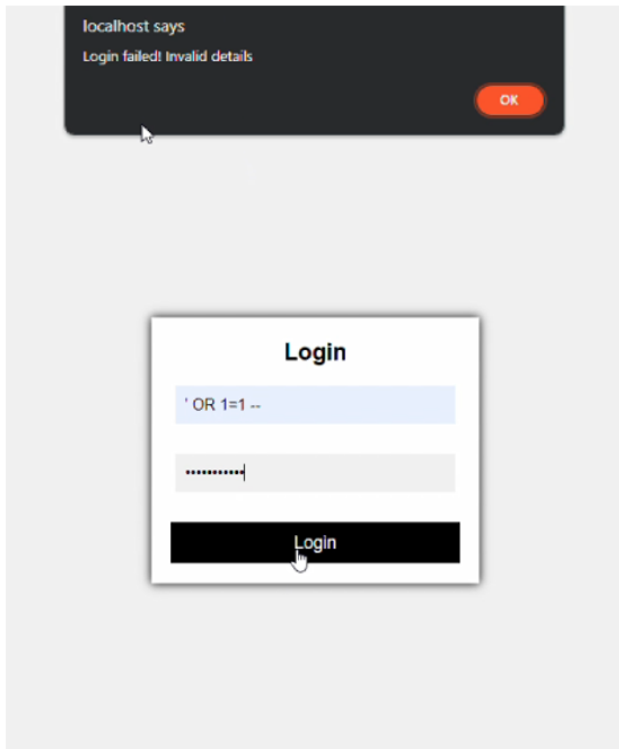
BLOCK DIAGRAM/ METHODOLOGY



CHAPTER-4

RESULTS AND DISCUSSION

SQL Injection



> Trying to perform SQL injection on page containing prepared statements results into denied access as Prepared statements separate the SQL query from the user input data. User input is treated as parameters, and it's bound to the prepared statement separately. The database server knows that these parameters are not part of the SQL query itself, preventing SQL injection attacks.

```
<?php
$host = "localhost";
$user = "root";
$password = "";
$database = "sanyam";
$conn = new mysqli($host, $user, $password, $database);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
if (isset($_POST['login'])) {
    $username = $_POST['username'];
    $password = $_POST['password'];
```

```

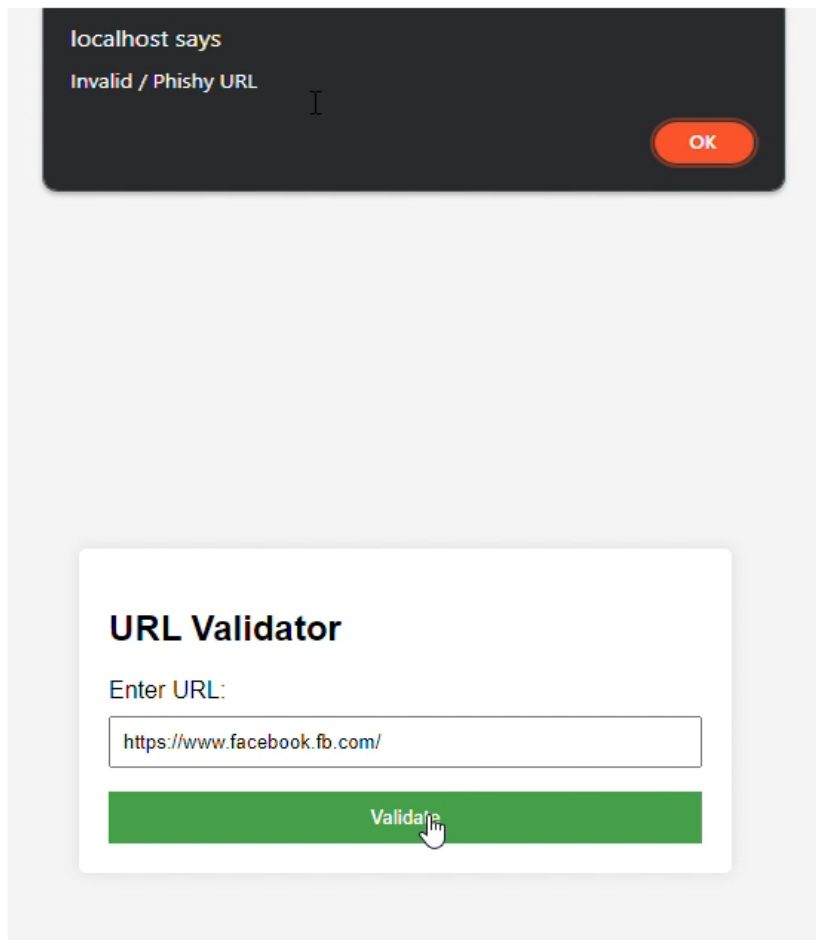
// Use a prepared statement with parameterized queries
$stmt = $conn->prepare("SELECT * FROM users WHERE username=? AND password=?");
$stmt->bind_param("ss", $username, $password);

if ($stmt->execute()) {
    $result = $stmt->get_result();

    if ($result->num_rows > 0) {
        echo "Login successful!";
    } else {
        echo "Login failed.";
    }
} else {
    echo "Login failed.";
}
// Close the prepared statement and database connection
$stmt->close();
$conn->close();
}
?>

```

Phishing



>Upon trying URL like <https://www.facebook.fb.com/>
The output “Invalid/phishing URL” will be received because the database does not have this URL and it does not match with the valid patterns.

```
function isPhishing(url) {  
  // Check for common phishing patterns in the URL  
  if (url.match(/\b(?:login|signin|account|password|secure)\b/i)) {  
    return true;  
  }  
  // Check if the URL uses IP address instead of domain name  
  var parsedUrl = new URL(url);  
  if (parsedUrl.hostname.match(/\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/)) {  
    return true;  
  }  
  // Check for unusual characters in the domain name  
  if (parsedUrl.hostname.match(/[^a-zA-Z0-9.-]/)) {  
    return true;  
  }  
  // Check if the URL uses an IP address in the path  
  if (parsedUrl.pathname.match(/\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/)) {  
    return true;  
  }  
}
```



```
// Additional checks based on specific criteria, such as subdomains, length, etc.
// If none of the checks match, consider it not phishing
return false;
}
function checkPhishing() {
    var urlInput = document.getElementById("urlInput").value;
    var resultElement = document.getElementById("result");

    if (isPhishing(urlInput)) {
        resultElement.innerHTML = "Phishing URL detected!";
    } else {
        resultElement.innerHTML = "Not a phishing URL.";
    }
}
}
```

Sniffing

Source	Destination	Protocol	Length	Info
192.168.0.4	18.192.172.30	TCP	66	60579 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
18.192.172.30	192.168.0.4	TCP	66	80 → 60579 [SYN, ACK] Seq=0 Ack=1 Win=62727 Len=0 MSS=1360 SACK_PERM=1 WS=128
192.168.0.4	18.192.172.30	TCP		
192.168.0.4	18.192.172.30	HTTP		
18.192.172.30	192.168.0.4	TCP		
18.192.172.30	192.168.0.4	TCP		
18.192.172.30	192.168.0.4	TCP		
192.168.0.4	18.192.172.30	TCP		
18.192.172.30	192.168.0.4	HTTP		like Gecko) Chrome/90.0.4430.212 Safari/537.36
192.168.0.4	18.192.172.30	TCP		Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
18.192.172.30	192.168.0.4	TCP		Referer: http://testphp.vulnweb.com/login.php
192.168.0.4	18.192.172.30	HTTP		Accept-Encoding: gzip, deflate
192.168.0.4	18.192.172.30	TCP		Accept-Language: en-US,en;q=0.9
192.168.0.4	18.192.172.30	HTTP		Cookie: login=test%2Ftest
18.192.172.30	192.168.0.4	TCP		login=test&pass=test HTTP/1.1 200 OK
18.192.172.30	192.168.0.4	TCP		Server: nginx/1.19.0
18.192.172.30	192.168.0.4	TCP		Date: Wed, 19 May 2021 06:35:18 GMT
192.168.0.4	18.192.172.30	TCP		Content-Type: text/html; charset=UTF-8
18.192.172.30	192.168.0.4	HTTP		Transfer-Encoding: chunked
192.168.0.4	18.192.172.30	TCP		Connection: keep-alive
192.168.0.4	18.192.172.30	TCP		X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
18.192.172.30	192.168.0.4	TCP		Set-Cookie: login=test%2Ftest
192.168.0.4	18.192.172.30	TCP		Content-Encoding: gzip

on wire (2312 bits), 289 bytes captured (2312 bits on interface)	
LinkIn_e6:0d:1a (c4:e9:0a:e6:0d:1a), Dst: Host	
Version 4, Src: 18.192.172.30, Dst: 192.168.0.4	
Protocol, Src Port: 80, Dst Port: 60579, Seq: 0	
Segments (2955 bytes): #191(1360), #192(1360)	
Protocol	
Wireshark · Follow HTTP Stream (tcp.stream eq 11) · Wi-Fi	
like Gecko) Chrome/90.0.4430.212 Safari/537.36	
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9	
Referer: http://testphp.vulnweb.com/login.php	
Accept-Encoding: gzip, deflate	
Accept-Language: en-US,en;q=0.9	
Cookie: login=test%2Ftest	
login=test&pass=test HTTP/1.1 200 OK	
Server: nginx/1.19.0	
Date: Wed, 19 May 2021 06:35:18 GMT	
Content-Type: text/html; charset=UTF-8	
Transfer-Encoding: chunked	
Connection: keep-alive	
X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1	
Set-Cookie: login=test%2Ftest	
Content-Encoding: gzip	
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">	
<html><!-- InstanceBegin template="/Templates/main_dynamic_template.dwt.php" codeOutsideHTMLOutsideIsLocked="false" -->	
<head>	
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">	
<!-- InstanceBeginEditable name="document title rgn" -->	

Figure: Wireshark Traffic analysis

We were able to fetch the login credentials easily with the help of network analyser as shown in the Figure.

Prevention against phishing can be achieved by using techniques like:

- Encryption:

Use encryption protocols (e.g., HTTPS, VPNs) to secure data in transit. This makes it difficult for attackers to decipher the content of intercepted packets.

- Segmentation:

Implement network segmentation to limit the scope of sniffing attacks. Divide the network into segments, and use firewalls to control traffic between segments.

- Network Monitoring:

Employ intrusion detection systems (IDS) and intrusion prevention systems (IPS) to detect and

prevent unauthorized sniffing activities.

- Secure Protocols:

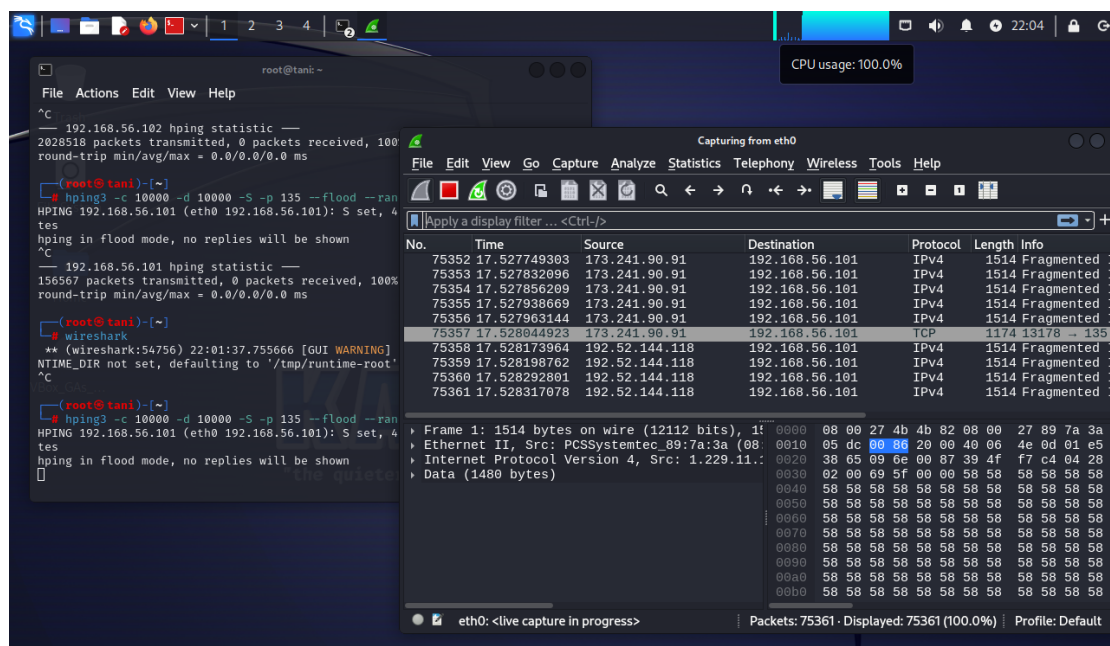
Where possible, use secure communication protocols (e.g., SSH instead of Telnet) to protect sensitive information.

Distributed Denial of Service

Demonstration of how to launch a SYN flood attack using a single computer and how it can be much more significant when launched from multiple machines. Also, how to define the number and size of packets to be sent to the target machine and how it affects CPU and network utilization.

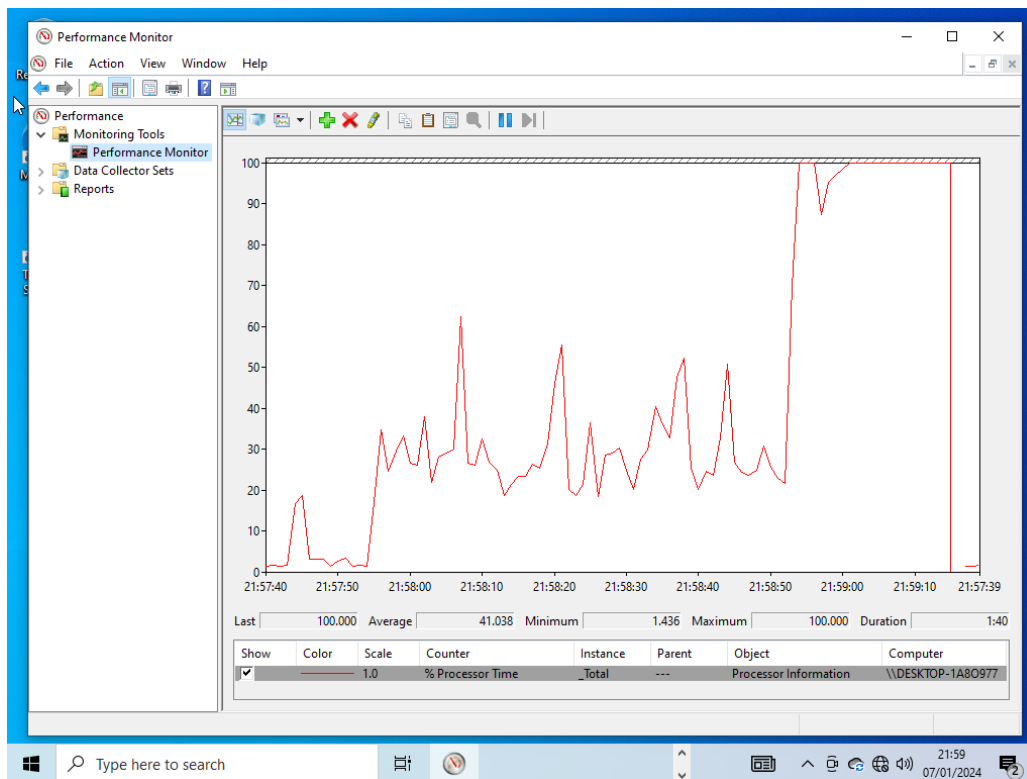
Usage of rand source flag to hide the identity of the attacker during a TCP SYN flood attack.

The demonstration is done by opening Wireshark to monitor the active Ethernet interface and running the hping3 command to show that the source IP address constantly changes when the rand source flag is used.

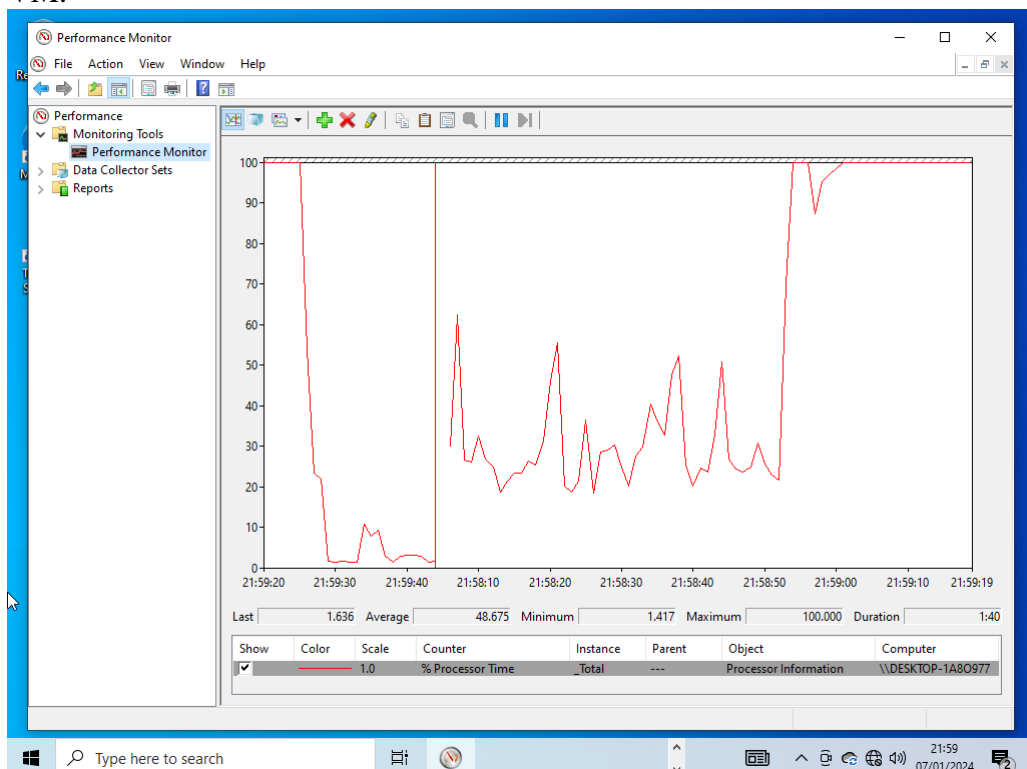


Monitoring the target machine's performance during an attack using the performance monitor and task manager.

The graph strikes up as soon as the flooding begins, even though there is no other applications running on it.



When we abort the command , we see an abrupt fall in the performance meter of the window's VM.



Cross Site Scripting [XSS]

Successful demonstration of XSS attack manipulating an image URL and then effectively closing off the source attribute using a double inverted comma to add a new attribute with malicious script code.


```

```

Your Message

Message Image

Send Message



data


The script is executed and the message is compromised. Now, this was a simple implementation on the user's side but, if the compromised message or image URL is stored in a database and rendered by multiple users, this way this simple attack can be fatal to the server.

127.0.0.1:5500 says
Hacked!

Your Message

Message Image

Send Message



data

some message

Chapter 5

CONCLUSION AND FUTURE WORK

Conclusion

In the culmination of this comprehensive study on web application attacks, encompassing a thorough investigation of attack models, detection methodologies, and prevention techniques, several key findings and insights emerge. The practical demonstrations of attacks like SQL injection, phishing, DDoS, and network sniffing provided a hands-on understanding of the potential threats faced by web applications in today's dynamic cybersecurity landscape.

The review of existing research papers and literature has illuminated the multifaceted nature of web application attacks. It is evident that the sophistication and diversity of these threats necessitate a multifaceted approach to cybersecurity. Detection mechanisms play a pivotal role in identifying and mitigating potential risks.

The assessment of various detection models has highlighted the strengths and limitations of each, emphasizing the importance of a holistic and adaptive detection strategy.

Furthermore, the exploration of prevention techniques underscores the proactive measures that can be taken to bolster web application security. Secure coding practices, input validation, robust authentication, access controls, and the implementation of web application firewalls were evaluated for their effectiveness in thwarting potential attacks. The project has established that a combination of these measures, tailored to the specific characteristics of the web application, contributes significantly to a robust defence against diverse attack vectors.

The practical demonstrations not only reinforced theoretical knowledge but also provided valuable insights into the real-world implications of web application vulnerabilities. The dynamic nature of cyber threats requires continuous adaptation and improvement of security measures. It is imperative for organizations to stay vigilant, update their security protocols, and educate their personnel on the latest threats and best practices.

Future Work

Phishing

Integration of Advanced Machine Learning Techniques: Researchers can explore the integration of advanced machine learning algorithms and techniques within the WC-PAD framework. This could involve the application of deep learning models, natural language processing (NLP) techniques, and advanced data analytics to improve the detection capabilities and overall accuracy of the phishing attack detection system.

XSS

Evaluation and Testing on Real-World Data: Conducting extensive evaluations and testing of the proposed techniques on a broader range of real-world web applications and datasets would provide valuable insights into their practical effectiveness and potential limitations. This could involve collaborating with various organizations to obtain diverse datasets and ensure that the proposed solutions are comprehensive and adaptable to different

SQL Injection

Investigating the use of automated tools and techniques for identifying and mitigating SQL injection vulnerabilities.

DDoS

Anomaly-Based DDoS Detection Algorithms: Investigate the use of advanced anomaly detection algorithms, such as deep learning and recurrent neural networks, to enhance the capability of DDoS detection systems in identifying previously unknown and sophisticated DDoS attack patterns.

Sniffing

Enhanced Intrusion Detection Techniques: Develop more sophisticated and robust intrusion detection systems (IDS) that can effectively identify and prevent various types of network attacks, including those that utilize packet sniffing techniques.

REFERENCES

- [1] Limei Ma, Yijun Gao, Dongmei Zhao, Chen Zhao. Research on SQL Injection Attack and Prevention Technology Based on Web. International Conference on Computer Network, Electronic and Automation (ICCNEA), 2019.
- [2] Nathezhtha.T , Sangeetha.D, Vaidehi.V . WC-PAD: Web Crawling based Phishing Attack Detection. International Carnahan Conference on Security Technology (ICCST), 2019.
- [3] P.Anu, Dr.S.Vimala. A survey on sniffing attacks on computer networks. International Conference on Intelligent Computing and Control (I2C2), 2017.
- [4] Shaimaa Khalifa Mahmoud, Marco Alfonse, Mohamed Ismail Roushdy, Abdel-Badeeh M. Salem .A Comparative Analysis of Cross Site Scripting (XSS) Detecting and Defensive Techniques . The 8th IEEE International Conference on Intelligent Computing and Information Systems (ICICIS), 2017.
- [5] Boyang Zhang, Tao Zhang. DDoS Detection and Prevention Based on Artificial Intelligence Techniques. 3rd IEEE International Conference on Computer and Communications, 2017.