VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CSE3501 Information Security Analysis and Audit

Slot : L29+L30

SQL INJECTION VISUALIZATION AND COMPARISON WITH OTHER APPLICATION LAYER ATTACKS

Submitted By :

1)Lakshit Manish Sanghrajka- 19BCB0031

2) Harshvardhan Mishra-19BCB0125

3) Samriddhi Agarwal -19BCB0127

Guided By :

Prof. Ruby D

School of Computer Science & Engineering

VIT University - India

FALL SEMESTER 2020-21

ABSTRACT

The Internet is quickly developing these days, as more business exercises are being mechanized and an expanding number of PCs are being utilized to store delicate data, the requirement for secure PC frameworks turns out to be pretty much self-evident. Today's interconnected world makes everyone more susceptible to cyber-attacks. A digital assault is a noxious and conscious endeavor by an individual or association to break the data arrangement of another individual or association.

Web attacks are the ones which occur on a website or web applications. Some of the online assaults are: XSS, Injection assaults, SQLi, Phishing, savage power, slowloris assault and some more Among these attacks, application layer attacks are frequently reported attacks. In these times, the data is highly vulnerable, and the attackers continuously find new ways to illegally retrieve them, as a result, the data should be protected from such attacks.

INTRODUCTION

Cross-Site Scripting (XSS) assaults are a sort of infusion, where malevolent contents are infused into in any case harmless and confided in sites. XSS assaults happen when an aggressor utilizes a web application to send malignant code, by and large as a program side content, to an alternate end client.

SQL injection is a web security weakness that permits an attacker to meddle with the inquiries that an application makes to its information base. It generally allows an attacker to view data that they are not normally able to retrieve. This may incorporate information having a place with different clients, or whatever other information that the actual application can get to. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior.

Slowloris is an application layer DDoS assault which utilizes halfway HTTP solicitations to open associations between a solitary PC and a designated Web server, then, keeping those affiliations open similarly as may be achievable, in like manner overwhelming and toning down the target.
Our venture targets giving definite correlation by performing fitting perception of these attacks based on amount of data loss, quantity of these attacks and its detection.

OBJECTIVES

- Demonstrate the whole process of SQLi attack from attack to prevention and retrieval on a vulnerable fintech website.
- Demonstration of other different application layer attacks.
- Demonstrating all three types of cross site scripting attacks.
- Demonstration of Slowloris attack.
- Comparison of SQLi with other attacks based on various parameters.
- Graphical analysis and visualization of the attacks and their effects.

LITERATURE SURVEY

S. No	Paper Title	Summary
1.	Impact of SQL Injection in Database Security (2020)	Here, Gupta, H. along with other authors have described how SQL injection attacks are one of the dangerous and harmful security problems or attacks. SQL injection attack is very dangerous for every other type of web based application extending an original way to deal with relieving SQL Injection Attacks in an information base. They have illustrated a technique or method to prevent SQLIA by incorporating a hybrid encryption in the form of Advanced Encryption Standard (AES) and Elliptical Curve Cryptography (ECC). In this examination paper an incorporated methodology of encryption technique is followed to forestall the information bases of the web applications against SQL Injection Attack. Incidentally if an invader gains access to the database, then it can cause severe damage and ends up retrieving data or information. So to forestall these kind of assaults a

		consolidated methodology is anticipated , Advanced Encryption Standard (AES) at login stage to forestall the unapproved admittance to the information bases and then again Circular Curve Cryptography (ECC) to encode the information base so that without the key nobody can get to the database information. With the increase in the use of web-applications, vulnerabilities have also increased drastically and how to combat such difficulties by employing new approaches. Thus securing databases from the SQL injection attack.
2.	**SQL Injection: Classification and Prevention (2021)**	**Aditya Rai along with other authors described that With the world moving towards digitalization, more applications and servers are online hosted on the internet, more vulnerabilities came out which directly affects an individual and an organization financially and in terms of notoriety as well. Out of those numerous weaknesses like Injection, Deserialization, Cross site prearranging and then some. Injection stands top as the most critical vulnerability found in the web application. Injection itself is a broad vulnerability as it further**

		consists of SQL Injection, Command injection, LDAP Injection, No-SQL Injection etc. In this paper we have investigated SQL Injection, various kinds of SQL infusion assaults, their makes and remediation grasp this assault
3.	**A Comprehensive Survey for Detection and Prevention of SQL Injection (2021)**	**In this paper, the built project aims to stop SQL injection attacks and make the database safer. This system is online, and there is no need for implementation. This can be gotten to from any area through the web. This framework uses SQL injection protection to keep the database and data safe. It will make specific accentuation on encoding Credit card information utilizing AES (Advanced Encryption Standard) procedure. The shop gets an installment so that all that will be gotten. The user pay card data is then stored in a database. The gadget additionally stores client data in a scrambled structure using AES encryptionThe framework is intended to keep away from SQL infusion exercises that can break into the data set and run it.**
4.	**A Study on SQL Injection**	**Mostly the intermediate layer is used to accept input from the user through web application. To build this**

	Attacks (2016)	**layer scripting languages are used. So to exploit the database attacker uses SQL Queries. To confuse this layer SQL Queries are reshaped by the attackers. In this paper, the focus was on those reshaped SQL Queries. The analysts have examined SQL Injection assault and its different avoidance and identification instruments utilized previously, then after the fact 2011. We can conclude that even there are more number of security measures developed there are also equal number of exploitation done**
5.	**A Survey on Cross Site Scripting attacks (2019)**	**In this paper, the researchers have studied a specific case of attack against web applications. They saw how the existence of cross-site scripting (XSS for short) vulnerabilities on web applications can involve a great risk for both the application itself and its users. They have additionally reviewed existing methodologies for the counteraction of XSS assaults on weak applications, examining their advantages and downsides. Whether dealing with persistent or non-persistent XSS attacks, there are currently very interesting solutions which provide interesting**

		approaches to solve the problem. Yet, these arrangements present a few disappointments, some don't give sufficient security and can be effortlessly avoided, others are entirely perplexing, to the point that they become unfeasible in genuine situations.
6.	**An Analytical Study on Cross-Site Scripting (2020)**	**Cross-Site Scripting, also called as XSS, is a type of injection where malicious scripts are injected into trusted websites. Right when toxic code, normally as program side substance, is imbued using a web application to an alternate end client, a XSS assault is said to have occurred. Flaws which allow success to this attack are remarkably widespread and occur anywhere a web application handles the user input without validating or encoding it. A study carried out by Symantec states that more than 50% of the websites are vulnerable to the XSS attack. Security engineers of Microsoft coined the term "Cross-Site Scripting" in January of the year 2000, powerless against the XSS assault. Security architects of Microsoft began the expression "Cross-Site Scripting" in January of the year 2000.. But even if it was coined in**

		the year 2000, XSS vulnerabilities have been reported and exploited since the start of 1990's, whose prey have been all the (then, at that point) tech-monsters like Twitter, Myspace, Orkut, Facebook and YouTube. Therefore this is known by the name of "Cross-Site" Scripting. This assault could be joined with different assaults, for example, phishing assault to make it more deadly yet it generally isn't required, since it is now incredibly hard to manage according to a client viewpoint in light of the fact that much of the time it looks exceptionally real as it's utilizing assaults against our banks, our shopping sites and not some phony noxious site

Table 1: Literature Survey

EXISTING SYSTEM

The existing system demonstrates either only the detection or prevention of one attack at a time. Existing systems do not support the comparative analysis between different types of application layer attacks and a comparative measure of the scope of loss of data if any of those attacks occurred.

PROPOSED SYSTEM

The proposed system will not just detect SQLi attacks but we will also implement and demonstrate the prevention techniques on a financial company from scratch. Not just that, but we also aim to visualize and compare the output that we will be getting with other major application layer attacks like Cross-site scripting (XSS) and slowloris attacks. A fintech website vulnerable to SQLi attacks will be created using PHP, front end technologies which will be connected to the database. The question string entered by the client will be gone to the server and the entire interaction from assault to location to anticipation and reaction will be illustrated. After this part, Comparative analysis with similar attacks along with Graphical analysis and Visualisation of the attacks and output will be done. This will include graphical comparison between these attacks based on several different factors like the scope of loss of data possible, time taken by prevention methods, vulnerabilities etc. Proposed defense mechanisms will reduce the risks for these kinds of attacks to a minimum level. One of the main goals of this project is to raise the awareness of the users of web applications by presenting a real simulation scenario of SQL injection attack. Furthermore, the goal of this project is to present the key defence mechanisms for detection and prevention of SQL injection attacks, along with how it's different from other major application attacks. The architecture diagram for this proposed system is shown in the following slide.

ARCHITECTURE DIAGRAM

1) Low level design (Detailed Design): Let us now have a look at each of these implemented attacks in this project, individually:
    a) Sql injection: SQL Injection (SQLi) is a type of an injection attack that makes it possible to execute malicious SQL statements. These statements control a database server behind a web application. Attackers can use SQL Injection vulnerabilities to bypass application security measures. They can go around authentication and authorization of a web page or web application and retrieve the content of the entire SQL database. They can also use SQL Injection to add, modify, and delete records in the database. Criminals may use it to gain unauthorized access to your sensitive data: customer information, personal data, trade secrets, intellectual property, and more. SQL injection usually occurs when you ask a user for input, like their username/userid, and instead of a name/id, the user gives you an SQL statement that you will unknowingly run on your database.

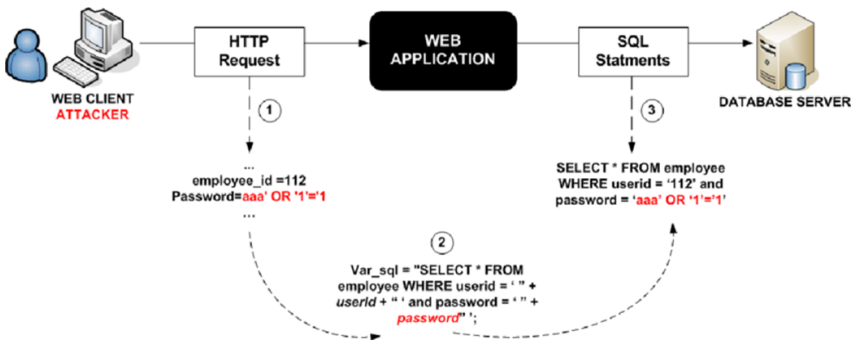

Fig.1: SQL Injection

b) Reflected Cross-site Scripting: Reflected XSS attacks, also known as non-persistent attacks, occur when a malicious script is reflected off of a web application to the victim's browser.

The script is activated through a link, which sends a request to a website with a vulnerability that enables execution of malicious scripts. The vulnerability is typically a result of incoming requests not being sufficiently sanitized, which allows for the manipulation of a web application's functions and the activation of malicious scripts.

To distribute the malicious link, a perpetrator typically embeds it into an email or third party website (e.g., in a comment section or in social media). The link is embedded inside an anchor text that provokes the user to click on it, which initiates the XSS request to an exploited website, reflecting the attack back to the user.

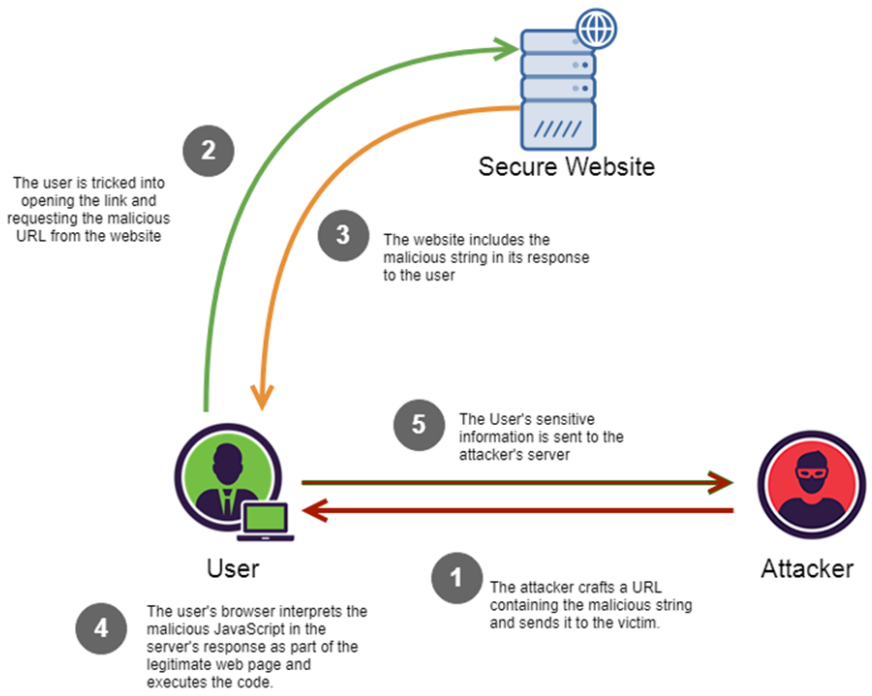

Fig.2: Reflected Cross Site Scripting

c) Stored Cross-site Scripting: To successfully execute a stored XSS attack, a perpetrator has to locate a vulnerability in a web application and then inject malicious script into its server (e.g., via a comment field).

One of the most frequent targets are websites that allow users to share content, including blogs, social networks, video sharing platforms and message boards. Every time the infected page is viewed, the malicious script is transmitted to the victim's browser.

Unlike a reflected attack, where the script is activated after a link is clicked, a stored attack only requires that the victim visit the compromised web page. This increases the reach of the attack, endangering all visitors no matter their level of vigilance.

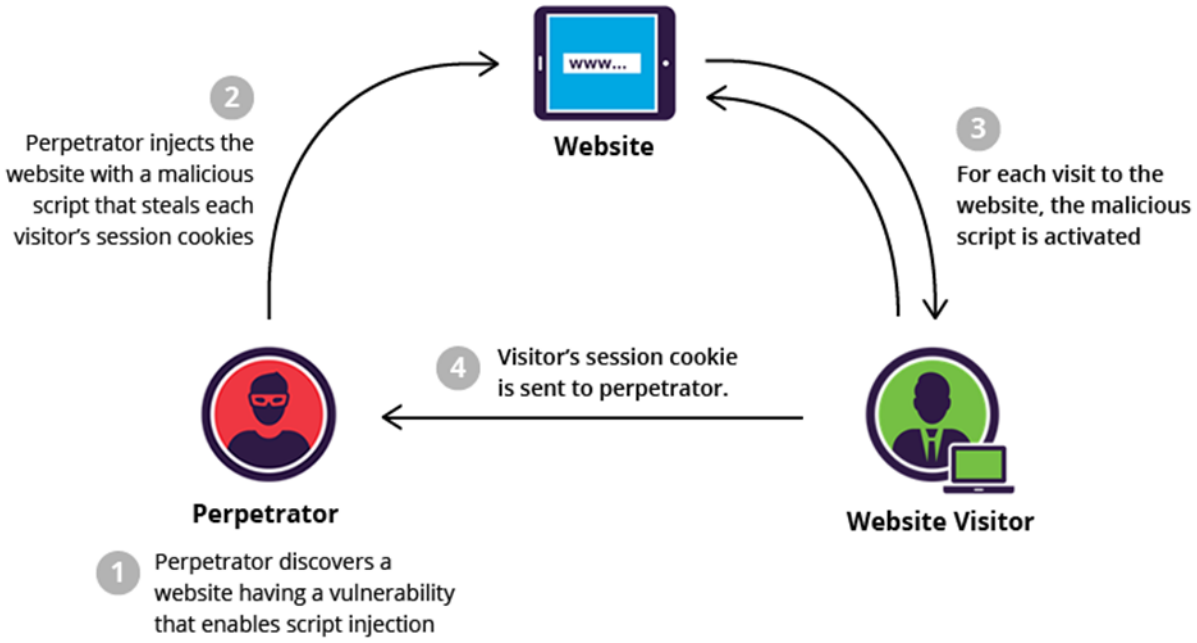

Fig.3: Stored Cross-Site Scripting

d) DOM based Cross-site Scripting: DOM-based cross-site scripting is a variant of both persistent and reflected XSS. In this attack, the malicious string is not actually parsed by the victim's browser until the website's legitimate JavaScript is executed. In the previous examples of persistent and reflected XSS attacks, the server inserts the malicious script into the page, which is then sent in a response to the victim. When the victim's browser receives the response, it assumes the malicious script to be part of the page's legitimate content and automatically executes it during page load as with any other script.

In the example of a DOM-based XSS attack, however, there is no malicious script inserted as part of the page; the only script that is automatically executed during page load is a legitimate part of the page. The problem is that this legitimate script directly makes use of user input in order to add HTML to the page. Because the malicious string is inserted into the page using innerHTML, it is parsed as HTML, causing the malicious script to be executed.

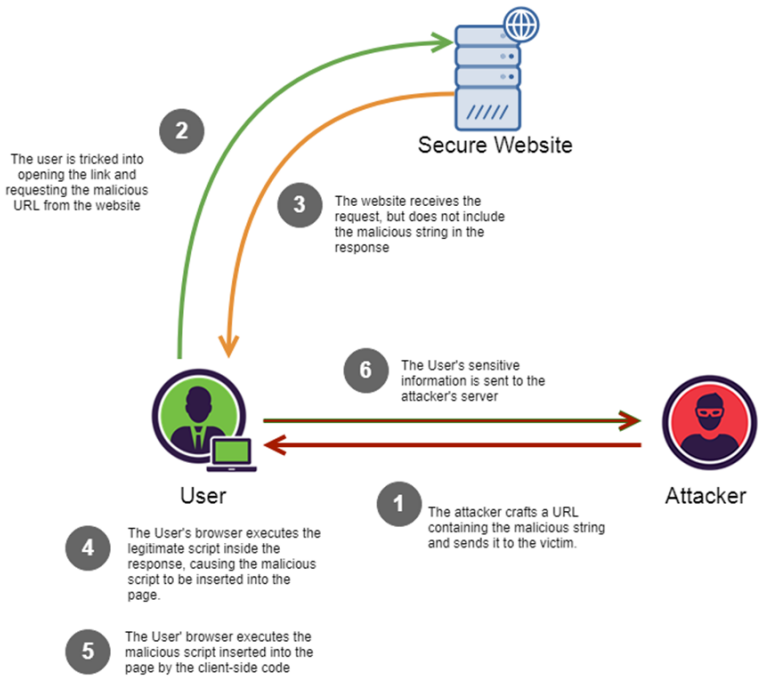

Fig.4: DOM based Scripting

e) Slowloris: Slowloris is an application layer DDoS attack which uses partial HTTP requests to open connections between a single computer and a targeted Web server, then keeping those connections open for as long as possible, thus overwhelming and slowing down the target.

This type of DDoS attack requires minimal bandwidth to launch and only impacts the target web server, leaving other services and ports unaffected. Slowloris DDoS attacks can target many types of Web server software, but has proven highly-effective against Apache 1.x and 2.x.

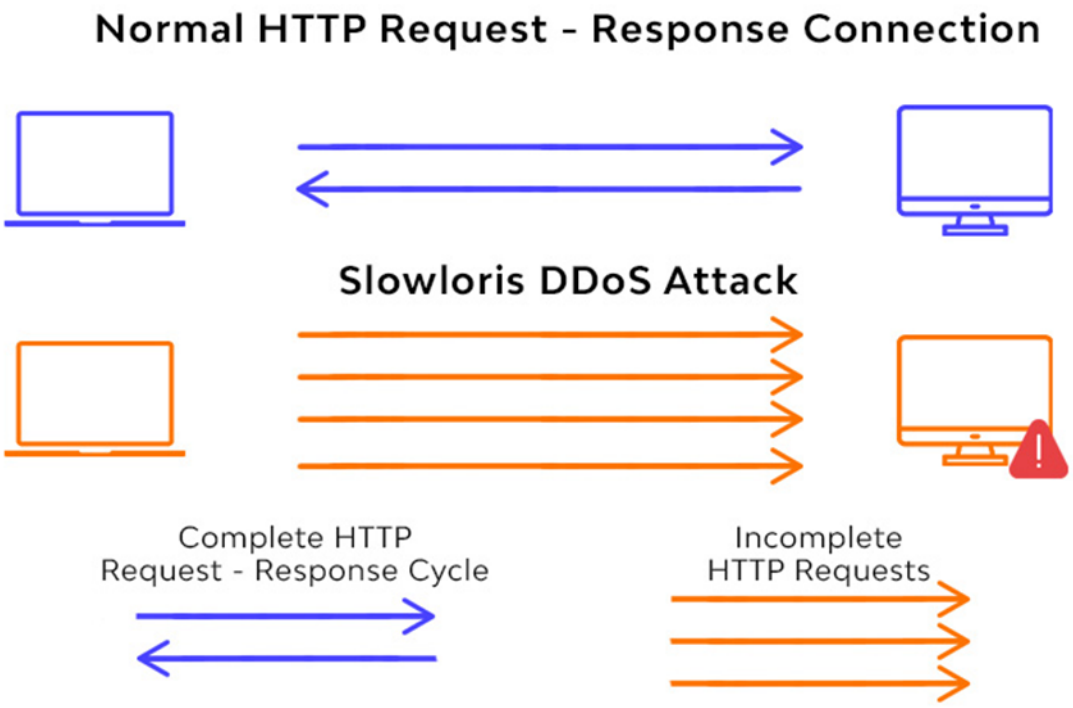

Fig.5: Slowloris attack

2) High level design (Black Box Design)

We have planned to demonstrate various application attacks and compare them based on various qualitative and quantitative measures. So, firstly, what are application layer attacks? An application-layer attack targets the layer of the internet that essentially

faces the end user. In the center of the architecture dig, we have an attacker who will be performing different application layer attacks using various tools and techniques.

Our first demonstrated attack is going to be sql injection, in which the attacker sends malicious SQL code or query for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details. For eg. an attacker sends a query like ‘ or 1=1 which always holds true and thus it can break through the firewall and attacker can receive the desired info from the db.

Second attack is Reflected cross site scripting it occurs when user input is immediately returned by a web application in an error message, search result, or any other response that includes some or all of the input provided by the user as part of the request, without that data being made safe to render in the browser, and without permanently storing the user provided data. Therefore it is a client side attack

Third one is Stored cross site scripting. It generally occurs when user input is stored on the target server, such as in a database, in a message forum, visitor log, comment field, etc. And then a victim is able to retrieve the stored data from the web application without that data being made safe to render in the browser. Therefore it is a server side attack

Fourth attack DOM based cross site scripting is a form of XSS where the source of the data is in the DOM, i.e data object model and the sink is also in the DOM, and the data flow never leaves the browser. For example, the source (where malicious data is read) could be the URL of the page (e.g., document.location.href), or it could be an element of the HTML, and the sink is a sensitive method call that causes the execution of the malicious data (e.g., document.write).”

Our last attack is a type of ddos attack , i.e slowloris. Slowloris attacks attempt to completely control the system resources by sending HTTP requests that never complete. Therefore, the web server waits indefinitely for requests, eventually consuming all its connection capacity. By exhausting TCP session availability, the server is frozen.

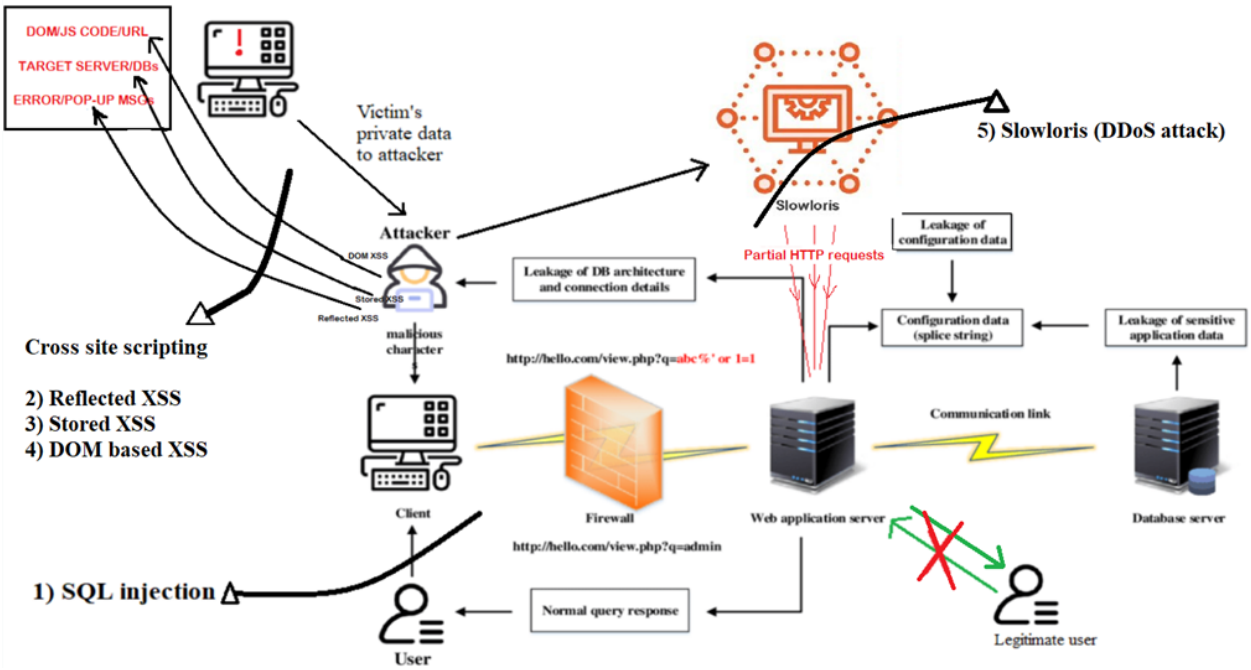

Fig.6: High level architecture

IMPLEMENTATION

1. Xss_DOM-low

```
<?php

# No protections, anything goes

?>
```

2. Xss_DOM-medium

```
<?php
```

```
// Is there any input?
if ( array_key_exists( "default", $_GET ) && !is_null ($_GET[ 'default' ]) ) {
    $default = $_GET['default'];

    # Do not allow script tags
    if (stripos ($default, "<script") !== false) {
        header ("location: ?default=English");
        exit;
    }
}

?>
```

3. Xss_DOM-impossible

```
<?php

# Don't need to do anything, protection handled on the client side

?>
```

4. Xss_Reflected-low

```
<?php

header ("X-XSS-Protection: 0");
```

```
// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Feedback for end user
    $html .= '<pre>Hello ' . $_GET[ 'name' ] . '</pre>';
}

?>
```

5. Xss_Reflected-medium

```
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = str_replace( '<script>', '', $_GET[ 'name' ] );

    // Feedback for end user
    $html .= "<pre>Hello ${name}</pre>";
}

?>
```

6. Xss_Reflected-high

```
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', $_GET[ 'name' ] );

    // Feedback for end user
```

```
    $html .= "<pre>Hello ${name}</pre>";
}

?>
```

7. Xss_Reflected-impossible

```
<?php

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $name = htmlspecialchars( $_GET[ 'name' ] );

    // Feedback for end user
    $html .= "<pre>Hello ${name}</pre>";
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

Similar amendments are made to security to XSS_Stored like XSS_Reflected according to corresponding security levels.

TOOLS AND TECHNIQUES USED

1. DVWA

2. SQL based Database
3. Hping3
4. Visme smart comparison chart maker
5. Frontend technologies - HTML, CSS, Bootstrap Basic
6. backend technology - Flask
7. Databases - Sqlite
8. System requirements - Any advanced operating system like windows, ubuntu, mac with an active internet connection

COMPARATIVE ANALYSIS

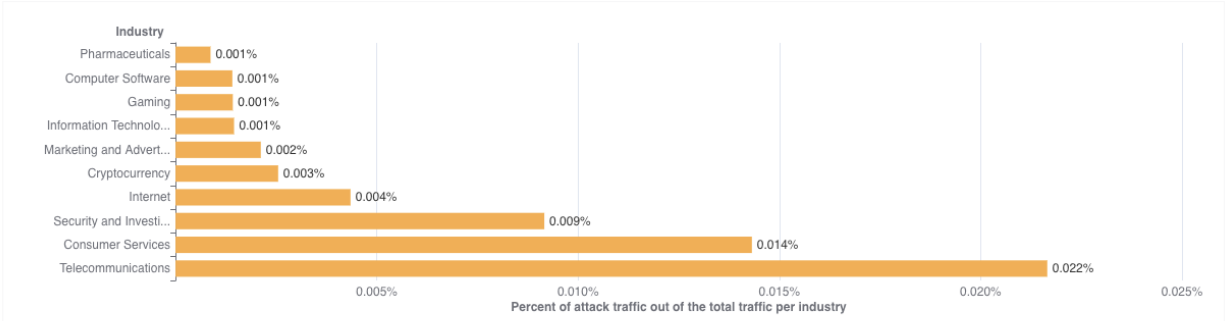

Fig. 7: DDoS activity per industry

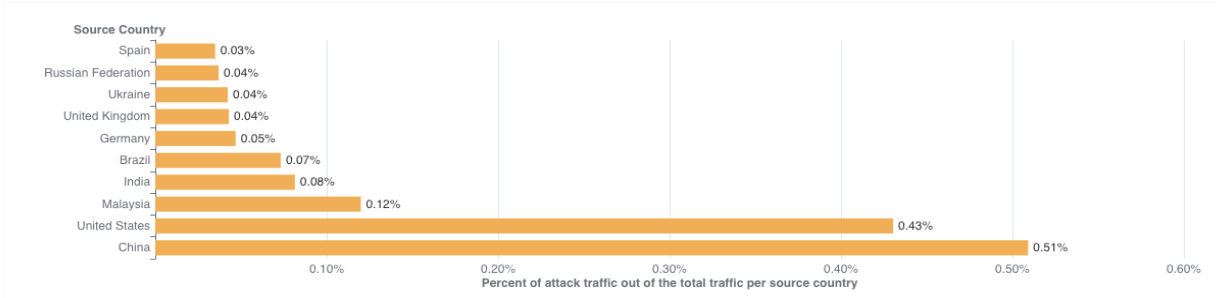

Fig. 8: DDoS activity by source country

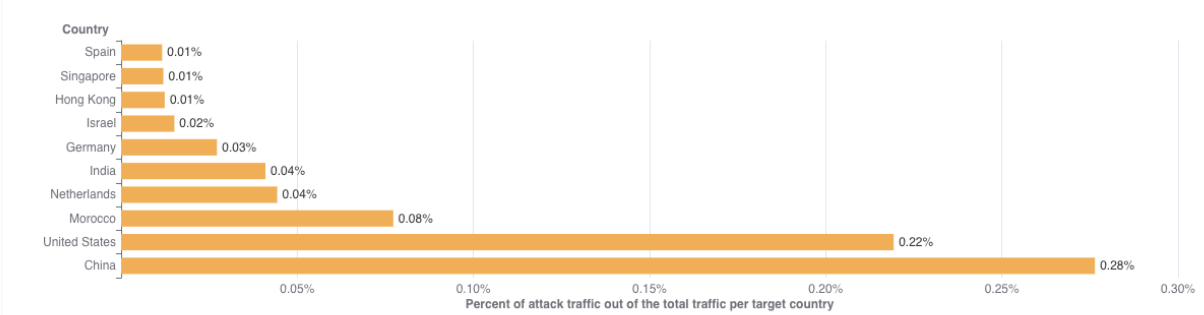

Fig. 9: DDoS activity by target country

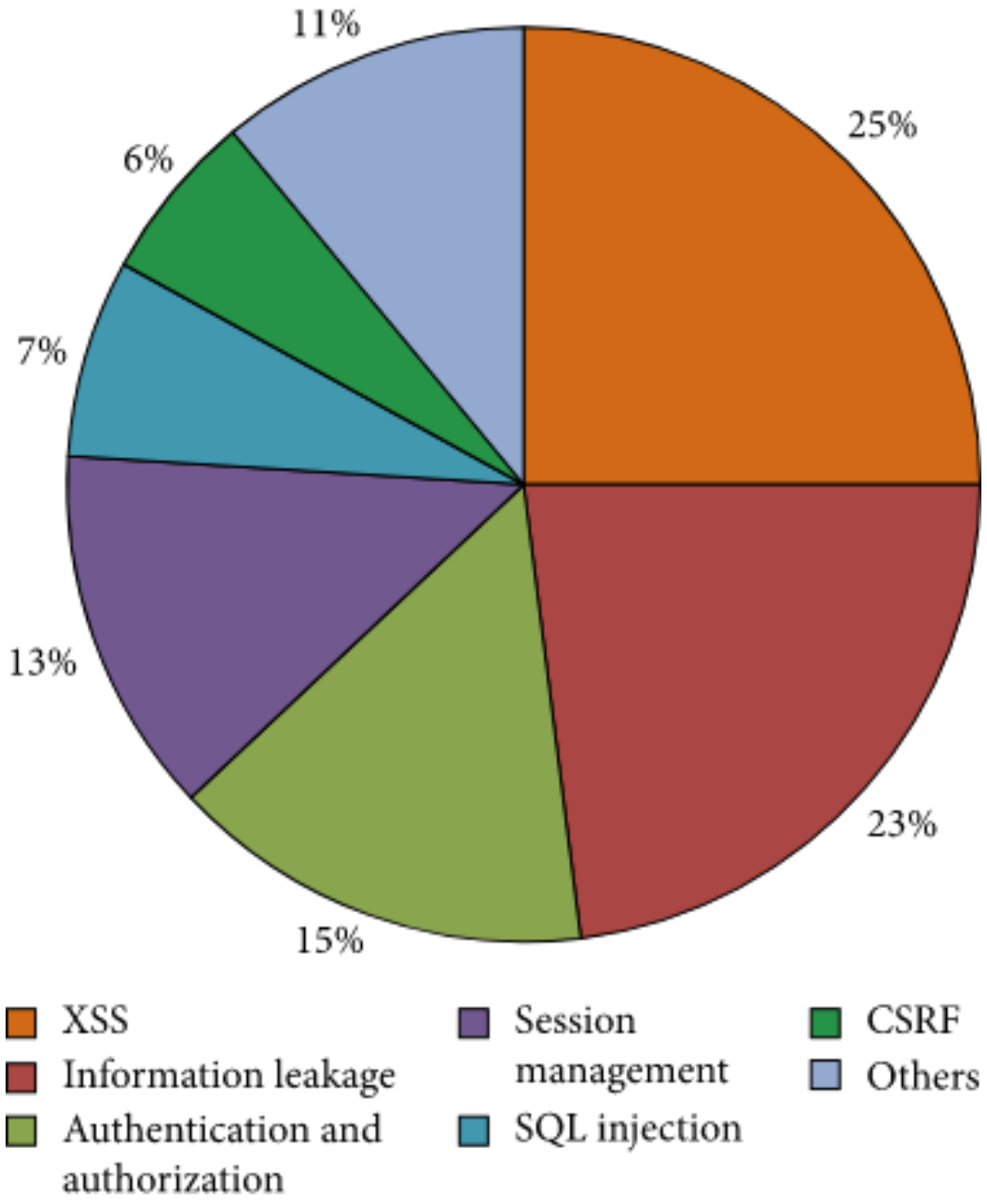

Fig. 10: Percentage of various attacks

RESULTS

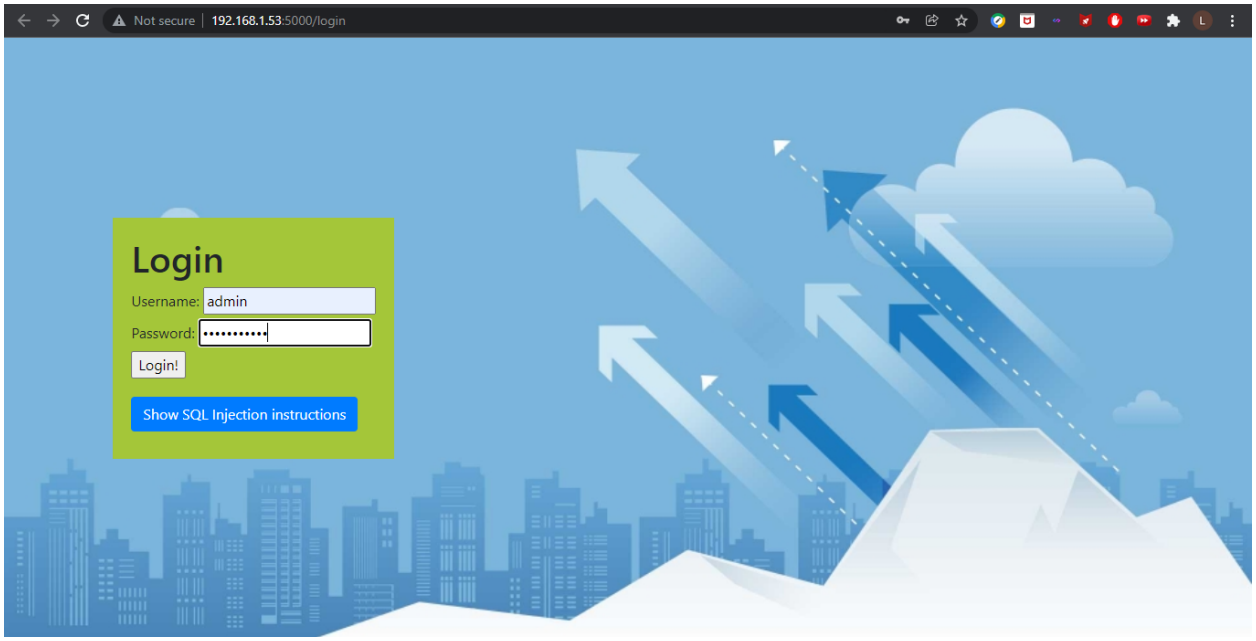

Fig. 11: Website user login page

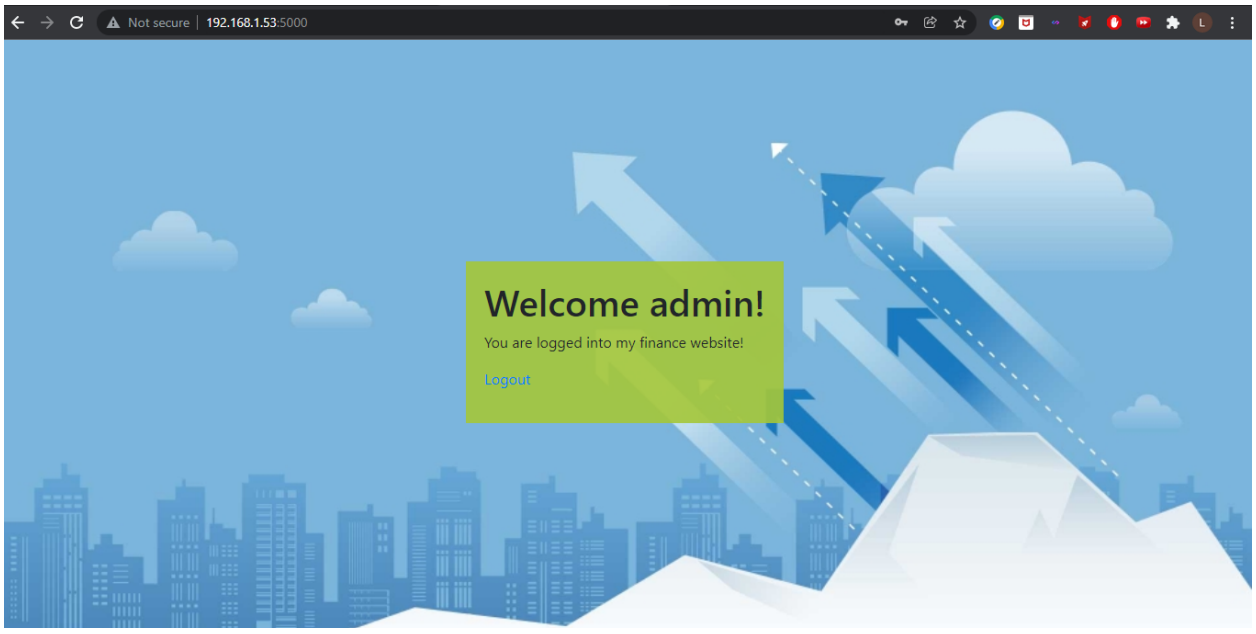

Fig. 12: User is successfully logged in

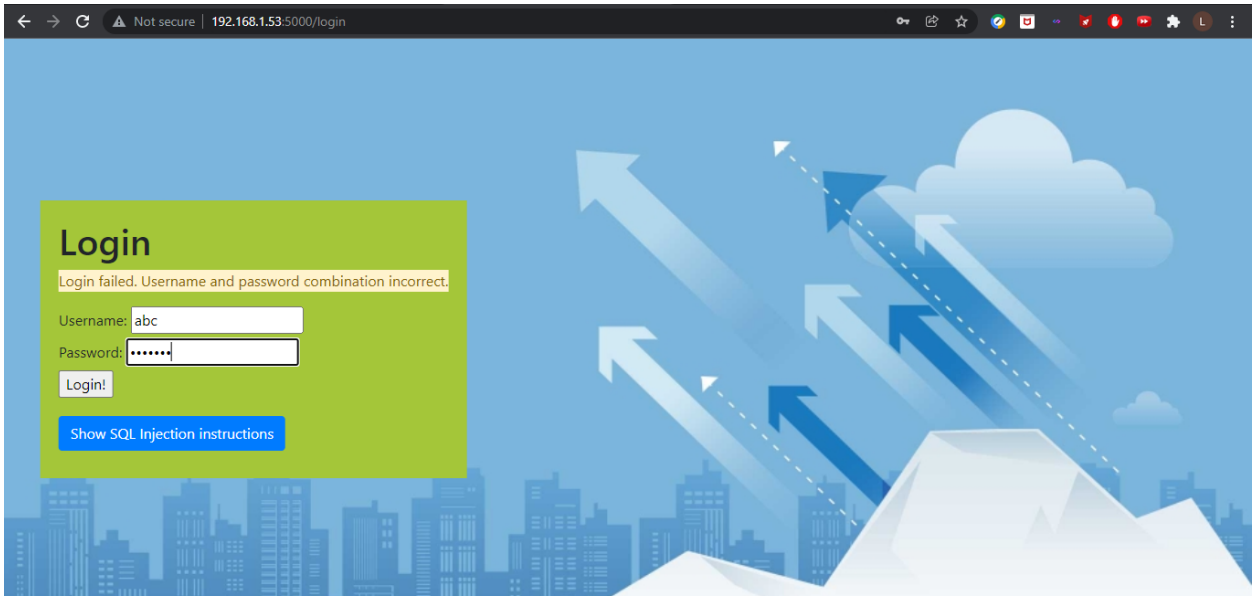

Fig. 13: Incorrect Login credentials gives the error

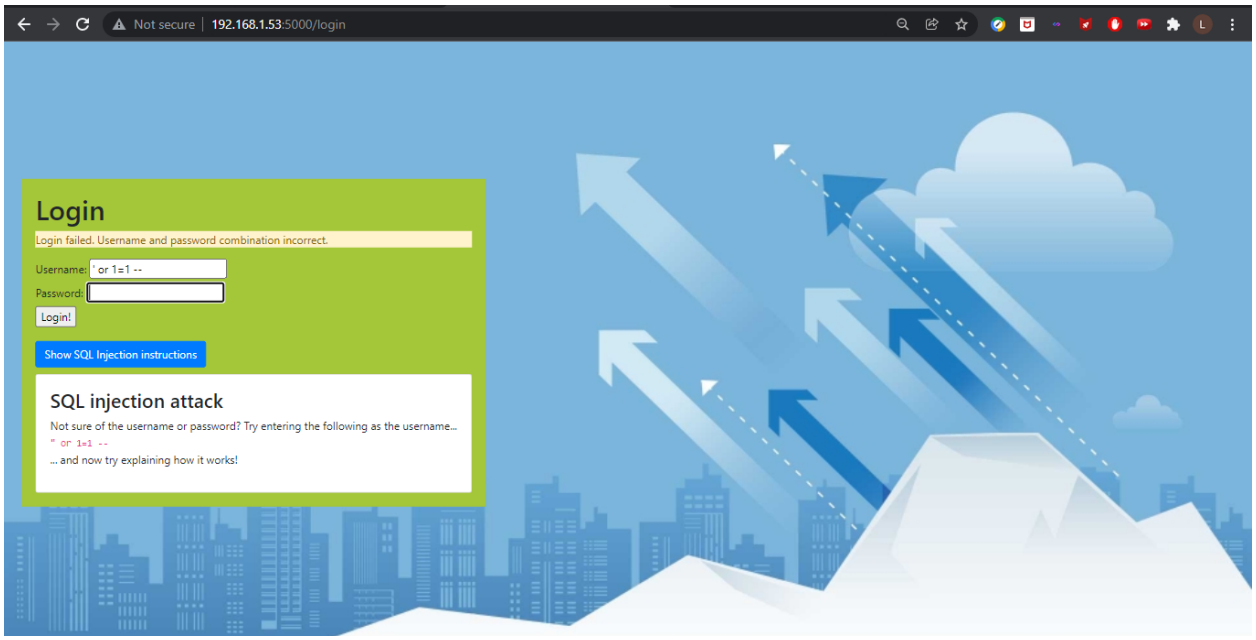

Fig. 14: Perform SQL injection query

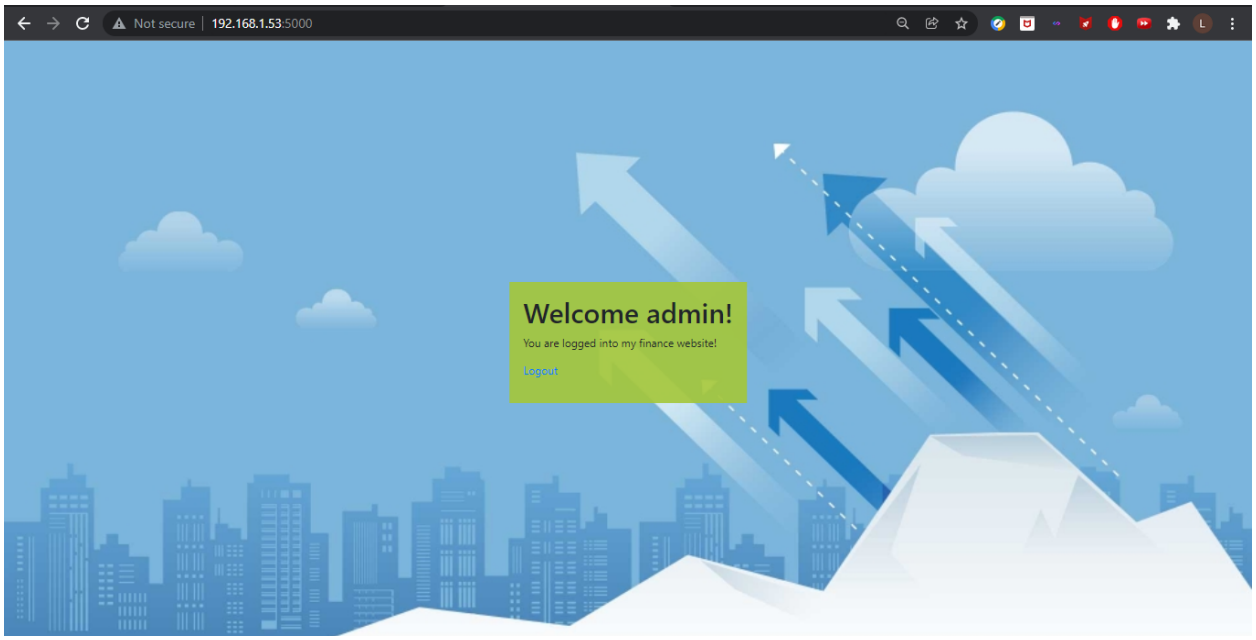

Fig. 15: Successfully getting logged in using SQL injection

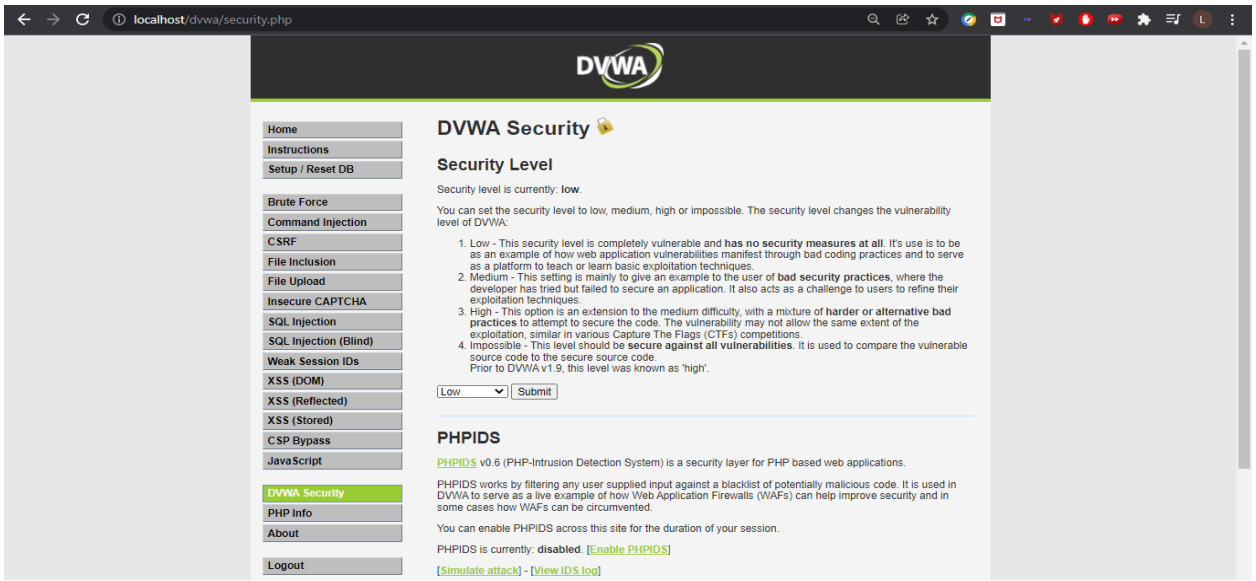

Fig. 16: Setting the security of DVWA

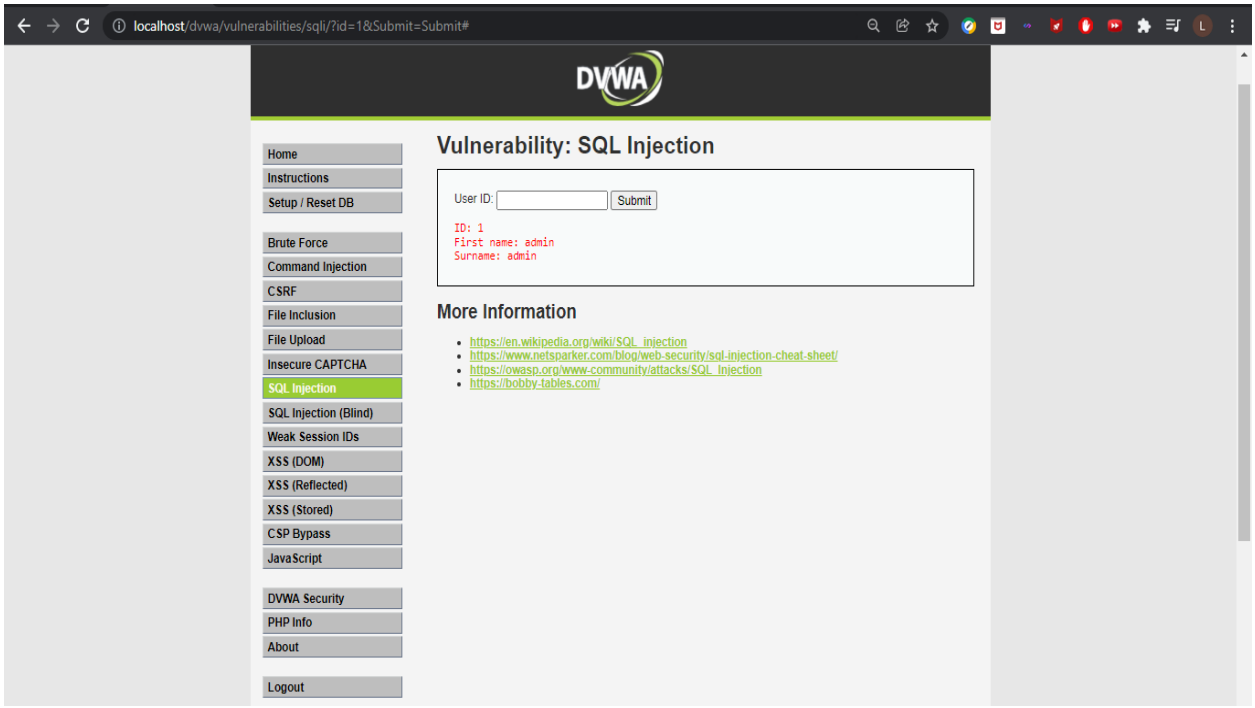

Fig. 17: Selecting the id for SQL injection

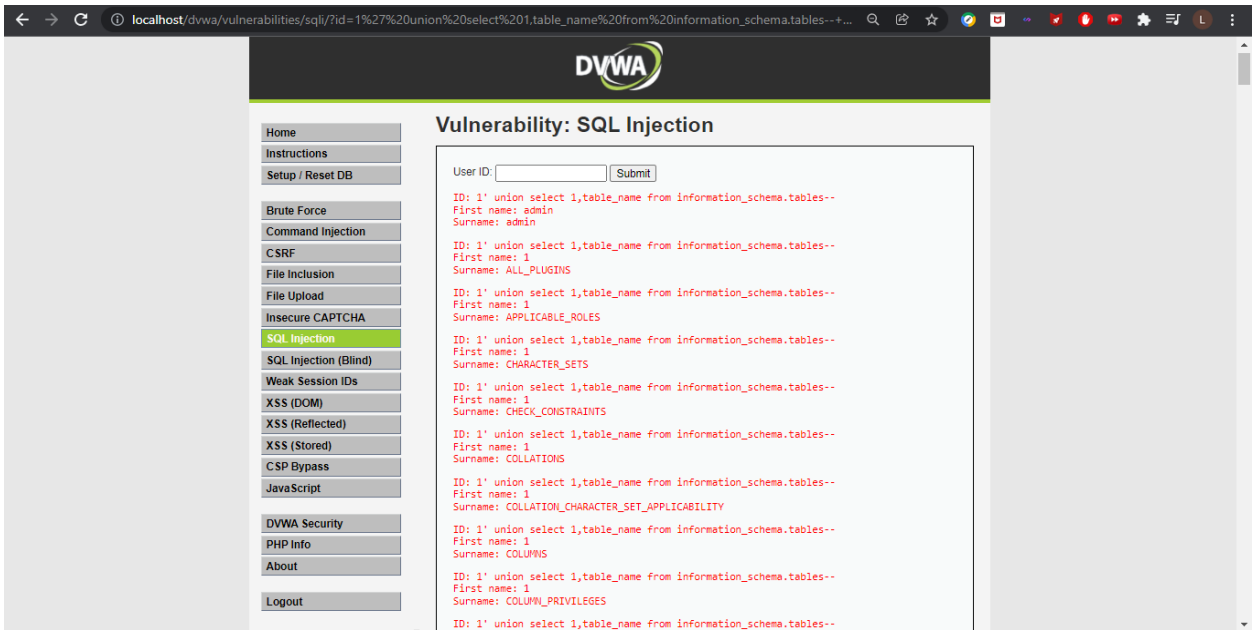

Fig. 18: Retrieving all the tables from the database

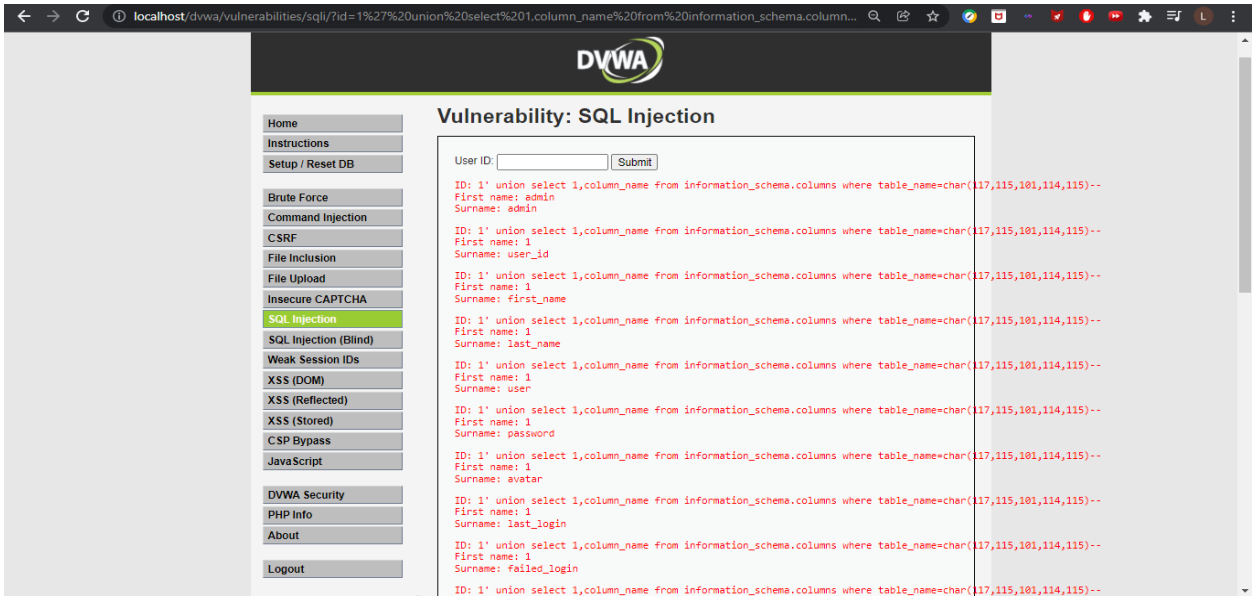

Fig. 19: Retrieving columns from the table user

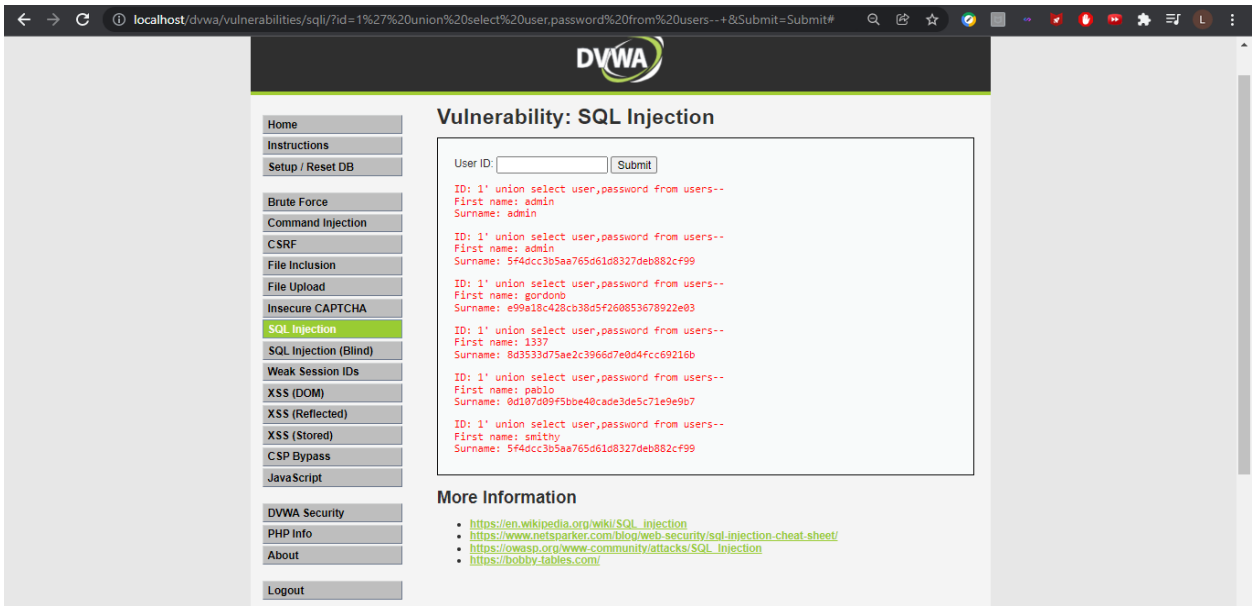

Fig. 20: Displaying Username and Password of users from table user

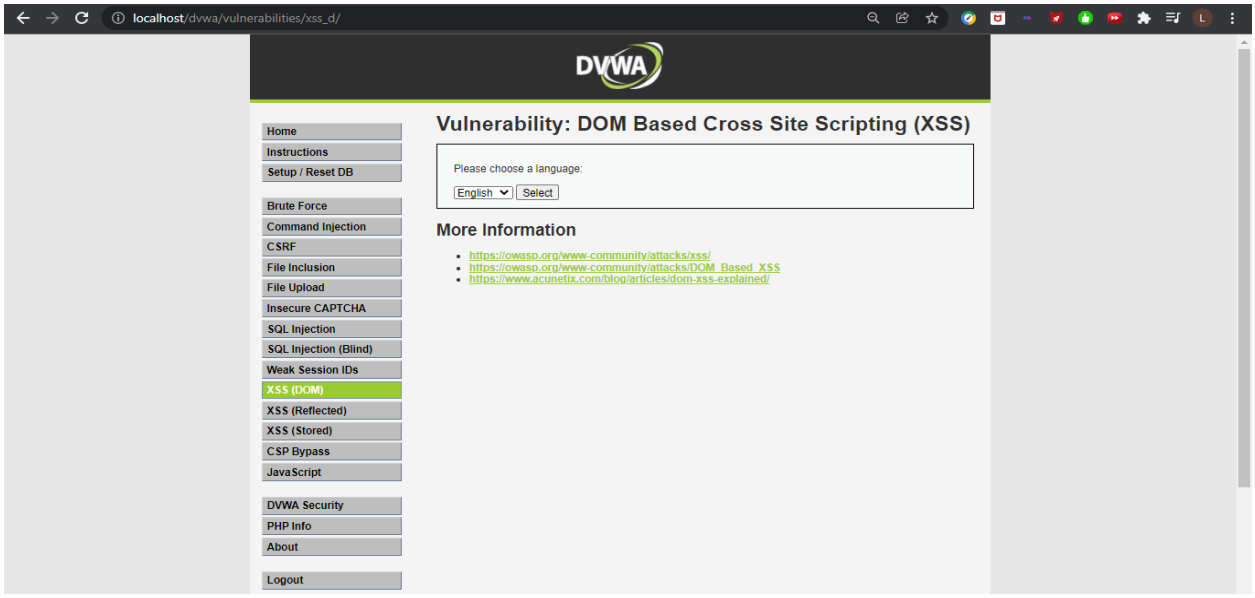

Fig. 21: DOM Based XSS

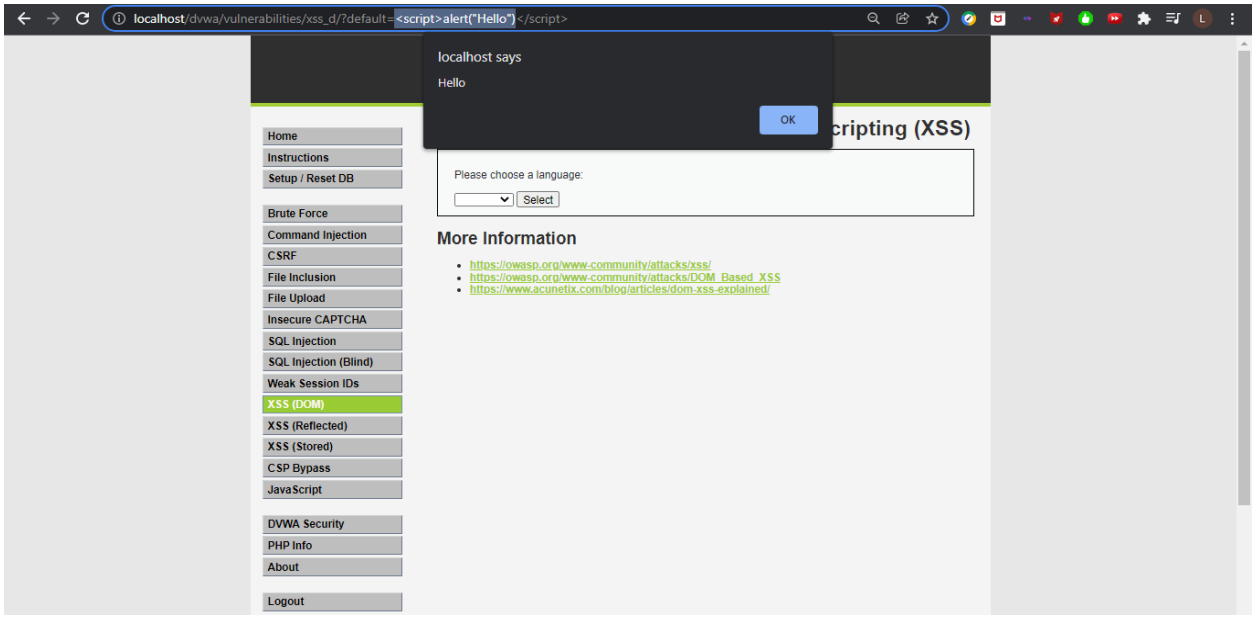

Fig. 22: Low level DOM XSS attack

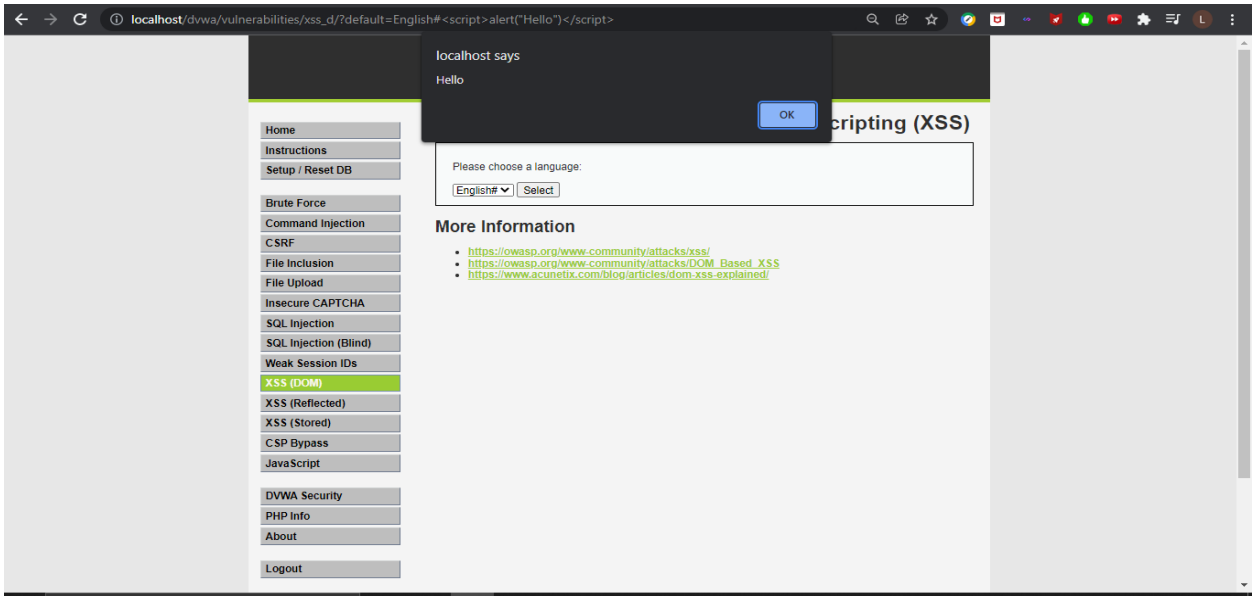

Fig. 23:Medium Level DOM XSS attack

Fig. 24: Site initially working fine before the traffic

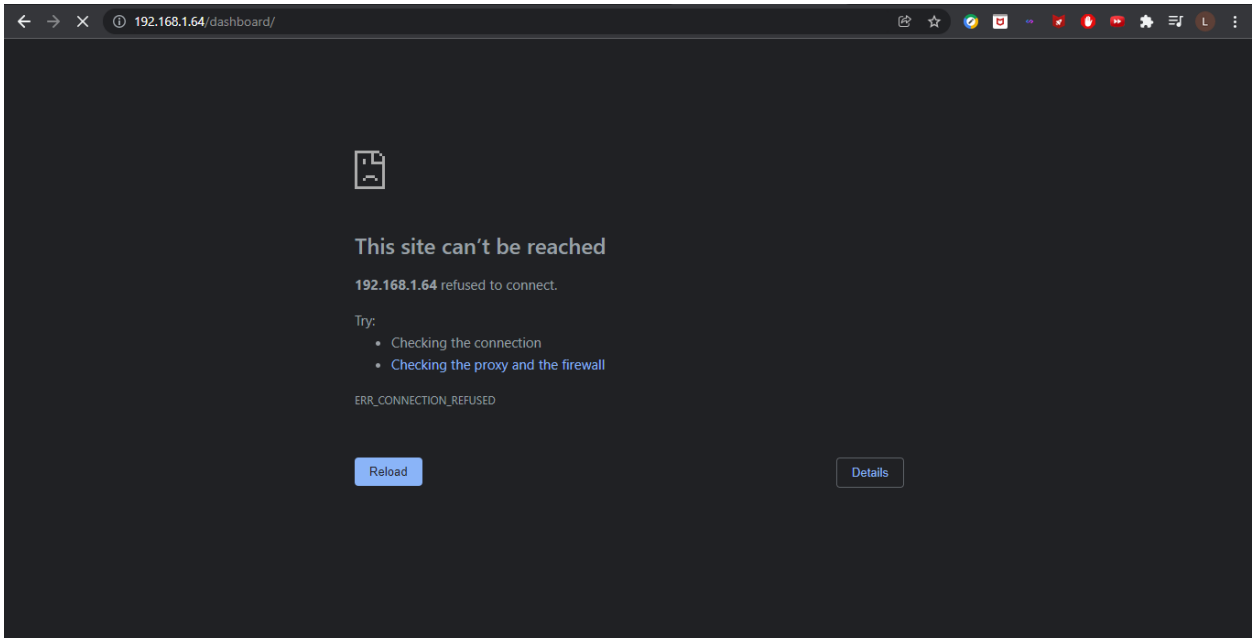

Fig. 25: Site is unreachable due to the traffic created by slowloris

```
C:\Users\Hii>slowloris -s 500 -p 80 192.168.1.64
[03-12-2021 09:36:04] Attacking 192.168.1.64 with 500 sockets.
[03-12-2021 09:36:04] Creating sockets...
[03-12-2021 09:36:07] Sending keep-alive headers... Socket count: 351
[03-12-2021 09:36:24] Sending keep-alive headers... Socket count: 351
[03-12-2021 09:36:41] Sending keep-alive headers... Socket count: 351
[03-12-2021 09:36:58] Sending keep-alive headers... Socket count: 351
```

Fig. 26: Slowloris is on and traffic is created on site

```
C:\Users\Hii>slowloris -s 500 -p 80 192.168.1.64
[03-12-2021 09:36:04] Attacking 192.168.1.64 with 500 sockets.
[03-12-2021 09:36:04] Creating sockets...
[03-12-2021 09:36:07] Sending keep-alive headers... Socket count: 351
[03-12-2021 09:36:24] Sending keep-alive headers... Socket count: 351
[03-12-2021 09:36:41] Sending keep-alive headers... Socket count: 351
[03-12-2021 09:36:58] Sending keep-alive headers... Socket count: 351
[03-12-2021 09:37:15] Sending keep-alive headers... Socket count: 351
[03-12-2021 09:37:32] Sending keep-alive headers... Socket count: 351
[03-12-2021 09:37:34] Stopping Slowloris
```

Fig. 27:Stopping Slowloris and site again starts working

CONCLUSION

- ➔ The use of application level attacks are serious cyber security vulnerabilities that are steadily growing and need to be addressed.
- ➔ The application layer is the hardest to defend. The vulnerabilities encountered here often rely on complex user input scenarios that are hard to define with an intrusion detection signature. This layer is also the most accessible and the most exposed to the outside world.
- ➔ Vann Abernethy of NSFOCUS states, “application layer attacks are potentially damaging to your critical infrastructure.”
- ➔ The reason why they are so damaging is because application level attacks can actually destroy or severely damage server, application, and database resources.

REFERENCES

1. H. Gupta, S. Mondal, S. Ray, B. Giri, R. Majumdar and V. P. Mishra, "Impact of SQL Injection in Database Security," 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), 2019, pp. 296-299, doi: 10.1109/ICCIKE47802.2019.9004430.
2. A. Rai, M. M. I. Miraz, D. Das, H. Kaur and Swati, "SQL Injection: Classification and Prevention," 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM), 2021, pp. 367-372, doi: 10.1109/ICIEM51511.2021.9445347.
3. A. Jana, P. Bordoloi and D. Maity, "Input-based Analysis Approach to Prevent SQL Injection Attacks," 2020 IEEE Region 10 Symposium (TENSYMP), 2020, pp. 1290-1293, doi: 10.1109/TENSYMP50017.2020.9230758.
4. Devi, Ruby & Venkatesan, R. & Koteeswaran, Raghuraman. (2016). A study on SQL injection techniques. International Journal of Pharmacy and Technology. 8. 22405-22415.
5. Garcia-Alfaro, Joaquin & Navarro-Arribas, Guillermo. (2009). A Survey on Cross-Site Scripting Attacks.
6. M. Singh, P. Singh and P. Kumar, "An Analytical Study on Cross-Site Scripting," 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA), 2020, pp. 1-6, doi: 10.1109/ICCSEA49143.2020.9132894.