Table Of Content

I. Introduction:

This project report, titled "Fortifying Web Applications: End-to-End Prevention and Solutions for Web Attacks," presents a comprehensive study on web application security conducted by a team of students from Netaji Subhas University of Technology. Under the supervision of Dr. Vipin Pal, the research aims to analyze trends in web attacks, identify vulnerabilities across different categories of web applications, and propose effective prevention strategies and solutions to enhance cybersecurity.

The study employs a thorough research methodology, drawing data from the Web Hacking Incident Database and various reputable security news sources. By examining attack frequencies, types, and their impact on different web application categories, the project offers valuable insights into the current state of web security. The report delves into the top five categories of web attacks, including SQL Injection, DDoS, Defacement, Account Hijacking, and Malware, and provides an in-depth analysis of their characteristics and prevention techniques. This research not only contributes to the understanding of web security challenges but also serves as a guide for developers to implement more robust security measures in future web applications.

II. Motivation:

As the number of web applications grows exponentially, the security gap widens, leading to increased risks of data breaches, DDoS attacks, and malware infections. This often results in severe consequences, such as financial losses, legal issues, and reputational damage. Data breaches, in particular, can expose sensitive user information, causing identity theft or fraud. Many websites, particularly those prioritizing fast deployment, often overlook essential security practices, leaving gaps that attackers exploit.

To combat these threats, developers and administrators must adopt a proactive, security-first approach. In an ever-evolving digital landscape, addressing security challenges early ensures better protection of both data and users.

The objective of the study is to find the trend of attacks on web applications and the target of attackers i.e. to know what vulnerabilities are commonly at risk and using that analysis to develop more secure web applications in future. We analyse top web attacks on different web categories. It is also an effort to guide web developers to take preventive measures during development by looking insight the attack trends in recent years.

III. Literature survey:

As the number of websites continues to grow exponentially, concerns over cybersecurity have failed to keep pace with this expansion. Many new and small-scale websites prioritize rapid deployment and user engagement over robust security measures, leaving them vulnerable to cyberattacks.Therefore, We need to implements security from the beginning process of development from Requirement Analysis to Design, Coding, Testing and Implementation but with the increasing complexity and interconnections in digital infrastructure, difficulty of achieving security is increasing exponentially.It was realized that security is required for each web application but the level of security may vary from organization to organization and the type of web application. Developers may choose the vulnerabilities to avoid for a web site, depending upon the trend that may better secure it in less time and efforts. This research work is an effort to look insight the security level requirements for various web categories by analysing attack trends on these categories. We have also studied abut top web attacks in deep.

IV. Problem Statement:

The rapid growth of web applications has led to a surge in security vulnerabilities, making them prime targets for attacks like data breaches, DDoS attacks, and malware infections. These incidents often result in severe financial, legal, and reputational consequences. This project aims to analyze recent trends in web application attacks, identify commonly exploited vulnerabilities across different web categories, and provide targeted security recommendations for developers. By understanding these attack patterns, the study seeks to guide the development of more secure web applications and mitigate future security risks.

V. Objective:

The objective of this project is to analyze attack trends on various web application categories to identify prevalent vulnerabilities and their impact. By understanding the unique security requirements of different web categories, the study aims to provide tailored security guidelines that developers can implement throughout the development lifecycle, from requirement analysis to deployment. This research seeks to enhance web application security by promoting a proactive, risk-based approach and exploring innovative solutions like machine learning to protect against evolving cyber threats.

VI. Simulation Platform and requirements:

The simulation platform for this project includes **OWASP WebGoat**, **DVWA (Damn Vulnerable Web Application)**, **Metasploit Framework**, and **Burp Suite**. **Kali Linux, Parrot Linux & BlackArch Linux** should be used for these hacking tools. These tools will be used to test and analyze common web security vulnerabilities such as SQL Injection and Cross-Site Scripting (XSS). Additionally, a custom web application with intentional vulnerabilities can be developed using Node.js or Java for specific attack simulations.

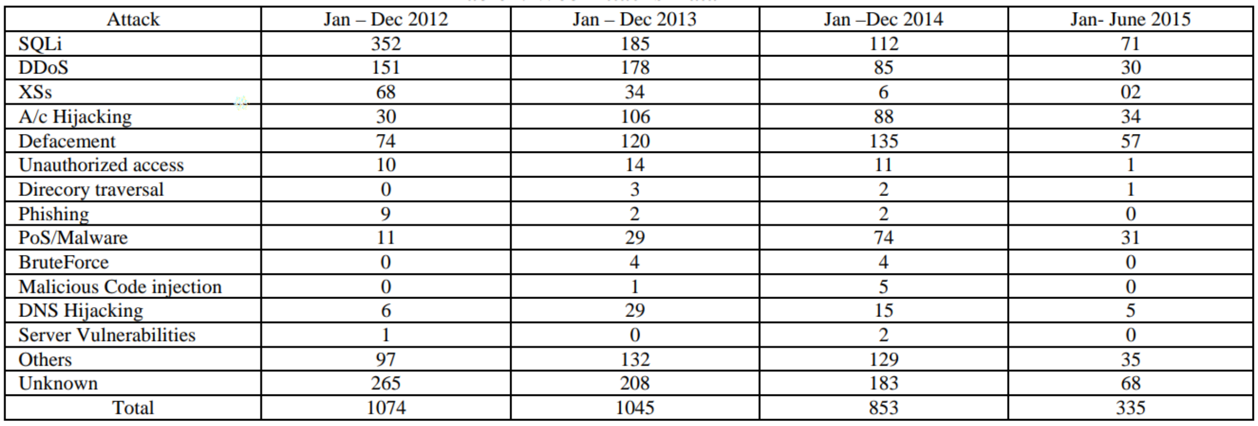
The simulation platform for the web application security analysis should utilize SQL databases and data visualization tools like **PowerBI** for effective data processing. It must support frameworks like **Django and ASP.NET**, and integrate with penetration testing tools such as **OWASP ZAP and Burp Suite** to assess vulnerabilities

Requirements include a development environment with Visual Studio Code or , web servers like Apache, and a database such as **MySQL** or **MongoDB**. Recommended hardware specifications are an **Intel Core i5 processor**, **8 GB RAM (16 GB preferred)**, and **100 GB of free disk space**. A virtual machine setup (e.g., VirtualBox) can be used for isolating the testing environment. This setup will enable comprehensive vulnerability analysis, helping developers build more secure web applications.

VII. Outcome Evaluation:

a) Web Attacks Analysis:

Frequencies of attacks and their respective frequency.



Attack	Jan – Dec 2012	Jan – Dec 2013	Jan –Dec 2014	Jan- June 2015
SQLi	352	185	112	71
DDoS	151	178	85	30
XSs	68	34	6	02
A/c Hijacking	30	106	88	34
Defacement	74	120	135	57
Unauthorized access	10	14	11	1
Direcory traversal	0	3	2	1
Phishing	9	2	2	0
PoS/Malware	11	29	74	31
BruteForce	0	4	4	0
Malicious Code injection	0	1	5	0
DNS Hijacking	6	29	15	5
Server Vulnerabilities	1	0	2	0
Others	97	132	129	35
Unknown	265	208	183	68
Total	1074	1045	853	335

D. Kaur and P. Kaur, "A Study of Web Application Security and Vulnerabilities," Procedia Computer Science, vol. 78, pp. 298-306, 2016.

strategies include input validation to sanitize user inputs, using parameterized queries to separate SQL logic from user data, implementing strict database access controls to limit user permissions, and employing monitoring and logging to detect unusual activity. Additionally, training machine learning models to identify suspicious SQL queries can enhance detection capabilities. While these techniques can mitigate the risk of SQLI, ongoing vigilance and adaptation are essential as attackers continuously evolve their methods.

b) XSS - Cross-Site Scripting

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user’s browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page. For more details on the different types of XSS flaws. We can prevent it by building code on some norms such as Variable Validation, Output Encoding and HTML Sanitization.

c) DOS/ DDOS

A Denial of Service (DoS) attack aims to disrupt the availability of a network resource by overwhelming it with a flood of illegitimate requests, causing the system to slow down or crash. A more sophisticated variant, the Distributed Denial of Service (DDoS) attack, involves multiple compromised devices, often part of a botnet, simultaneously targeting a single server or network. This coordinated effort generates an enormous volume of traffic, making it challenging to differentiate between legitimate and malicious requests. DoS and DDoS attacks can take various forms, such as volume-based attacks that flood the target with high traffic, protocol attacks that exploit weaknesses in network protocols, and application layer attacks that target specific services.DoS attacks can exploit memory buffer overflow or saturate server capacity with excessive packets. Signs of a DoS attack include sluggish behavior, system crashes, and overwhelmed server capacity. The impact of these attacks can be severe, leading to service disruptions, financial loss, and reputational damage. Effective mitigation strategies include implementing rate limiting, using web application firewalls (WAF) and intrusion prevention systems (IPS), and employing DDoS protection services like content delivery networks (CDNs) and cloud-based security solutions.

d) Session hijacking

Session hijacking, also known as cookie hijacking, is a cyber attack where an attacker gains unauthorized access to a user's session by stealing or manipulating session cookies or tokens. These cookies contain session identifiers, allowing attackers to impersonate the user and gain access to their account without needing login credentials. This can lead to severe consequences, such as data theft, unauthorized transactions, or changes to account settings. Common methods of session hijacking include cross-site scripting (XSS), packet sniffing, and man-in-the-middle attacks. To mitigate session hijacking, it is essential to use secure communication protocols like HTTPS, implement session timeouts, and use secure, HTTP-only, and encrypted cookies. Ensuring robust session management and security measures helps protect user sessions from unauthorized access.

e) Man-in-the-Middle

A Man-in-the-Middle (MITM) attack occurs when a malicious actor intercepts and potentially alters communication between two parties without their knowledge. This enables the attacker to eavesdrop, manipulate, or steal sensitive information, such as login credentials or financial data. MITM attacks can be executed in various ways, such as by compromising a Wi-Fi network, exploiting vulnerabilities in network protocols, or using phishing techniques to trick users into connecting to a malicious network. The consequences can be severe, leading to identity theft, financial loss, or unauthorized access to confidential data. To prevent MITM attacks, it is essential to use secure communication protocols like HTTPS, employ strong encryption, and verify the authenticity of certificates and connections. Awareness and vigilance are crucial in protecting against this type of threat.

f) Extension Vulnerability

Extension vulnerabilities occur when browser extensions, plugins, or add-ons, which are meant to enhance functionality, introduce security risks due to flaws in their design or implementation. Malicious or poorly developed extensions can be exploited to gain unauthorized access to sensitive data, such as browsing history, passwords, or personal information. They may also inject malicious scripts, alter website content, or perform actions on behalf of the user without their consent. These vulnerabilities can be particularly dangerous as extensions often have high levels of permission within the browser, making them an attractive target for attackers. To mitigate extension vulnerabilities, users should only install extensions from trusted sources, regularly update them, and review their permissions. Developers should follow secure coding practices and conduct thorough security testing to prevent potential exploits.

IX. Prevention & Solutions:

To prevent or mitigate the web attacks you've chosen, here are some techniques and algorithms you can implement on the developer's end:

a) SQL Injection (SQLi):

Various techniques have been developed to prevent SQLI attacks. Machine learning and hybrid approaches have emerged as powerful tools, utilizing algorithms such as Naive Bayes, Decision Trees, Support Vector Machines, and Neural Networks to detect and prevent attacks. These methods analyze features from web logs, cookies, session usage, and HTTP(S) requests to identify potential threats. Other prevention techniques include input validation and sanitization, which involve filtering and cleaning user inputs to remove potentially malicious code. Parameterized queries and prepared statements are also effective, as they separate SQL code from user input, making it difficult for attackers to inject malicious queries. Web application firewalls (WAFs) can provide an additional layer of security by filtering out suspicious requests. Encryption of sensitive data and implementing least privilege principles in database access further enhance security. Regular security audits and penetration testing help identify and address vulnerabilities before they can be exploited.

b)XSS (Cross-Site Scripting):

To prevent XSS attacks, it is crucial to use input validation and proper escaping techniques. Input validation ensures that only expected and safe data is accepted from users, blocking harmful characters like < or > and returning an error message if invalid input is detected. Escaping involves converting user input to a safe format before displaying it in web pages, preventing it from being executed as code. Using functions like htmlspecialchars() or htmlentities() helps in this process by converting special characters into HTML-safe equivalents. Additionally, implementing a Content-Security-Policy (CSP) can further enhance security by instructing the browser to only trust resources from a predefined whitelist, blocking any untrusted sources. Together, these strategies effectively mitigate the risk of XSS attacks by ensuring that user input cannot be used to manipulate web page behavior or inject malicious scripts.

c)DOS/DDOS (Denial of Service):

To prevent and mitigate DoS/DDoS attacks in your final year project, consider implementing multiple layers of defense. Start by using a Web Application Firewall (WAF) to filter and monitor incoming traffic, blocking suspicious activity. Rate limiting is another crucial strategy, as it restricts the number of requests a user can make in a given timeframe, preventing overwhelming spikes. You can also use load balancers and Content Delivery Networks (CDNs) to distribute traffic across multiple servers, reducing the impact of an attack on a single server. Additionally, configuring your server to automatically detect and respond to unusual traffic patterns helps in early detection and mitigation. For more advanced protection, consider implementing DDoS protection services like Cloudflare or AWS Shield, which can absorb and deflect large-scale attacks. Finally, ensure your project's infrastructure is scalable, so it can handle unexpected traffic surges without compromising performance or availability.

d) Session Hijacking:

To prevent the user from session hijacking use Strong Session ID to avoid hijacked or deciphered. SSL and SSH provide strong encryption using SSL certificate. There must be a log out function for every session termination, login for regeneration of Session ID. HTTPS connection should be used for passing authentication cookies and also reduce the life span of session or cookie. Session hijacking can be prevented at the user level by clear the history, offline contents, and cookies from the browser after every sensitive transaction. To protect from session hijacking there are different tools and techniques are available. By using a sniffer on network attacker can be detected. ANTI-SNIFF-It can detect any sniffer on the network used to capture packets

e)Man-in-the-Middle (MITM) Attacks:

To prevent DNS spoofing ensure that latest version of DNS software with recent security patches is installed. Also Ensure that auditing is enabled on all DNS server. Most popular email services and online banking applications rely on HTTPS to ensure that communications between our web browser and their servers is in encrypted form. By using DH for key generation and Blowfish for encryption will enhance data security over SSL and HTTPS. ARP poisoning can be avoided by running shell script at the backend. This will keep track of entries in the ARP cache table. Different security measures can be used such as operating systems onto the network should be upgraded, network designing from security point of view, network devices and the computers onto the network should be updated periodically and the patches should be installed regularly. Moreover we should verify SSL certificates and pgp keys should be used.

f) Extension Vulnerability:

A new browser extension system can be used to protect browser from this attack. Extensions run with least privileges can be exploited by a malicious website to avoid divide extension into three components: content scripts, extension core, and native library. An attacker would need to convince the extension to forward malicious input from the content script to the extension core and from the extension core to the native binary to gain users full privileges. Different components of an extension are isolated from each other by strong protection boundaries: each component runs in a separate operating system process. The content script and the extension core run in sandboxed processes, they cannot use operating system services. The content script is isolated from its associated web page by running in a separate JavaScript heap but both uses the same DOM, prevents JavaScript capability leaks.

X. Conclusion:

This study highlights the importance of securing web applications by addressing the most common attack vectors such as SQL Injection, DDoS, Defacement, Account Hijacking, and Malware. Through the analysis of attack trends across different web application categories, the study emphasizes the need for tailored security measures based on the unique vulnerabilities and risks faced by each category. By implementing proactive prevention techniques and incorporating modern solutions like machine learning, developers can greatly improve the security of web applications, minimizing the risk of breaches and enhancing user protection.

Ultimately, the research underlines that while perfect security may not be attainable, significant strides can be made by embedding security into every phase of the development lifecycle. By staying informed about evolving attack trends and leveraging emerging technologies, developers can build more resilient applications that are better equipped to safeguard both data and users in an increasingly complex digital environment

XI. References:

1. D. Kaur and P. Kaur, "A Study of Web Application Security and Vulnerabilities," Procedia Computer Science, vol. 78, pp. 298-306, 2016.
2. Demilie, W.B., & Deriba, F.G., "Detection and prevention of SQLI attacks and developing compressive framework using machine learning and hybrid techniques," Journal of Big Data, 9, Article 124, 2022.
3. Patil, S. S., & Chavan, R. K., "Web Browser Security: Different Attacks Detection and Prevention Techniques," International Journal of Computer Applications, 170(9), 35-41, 2017.
4. "Application Vulnerability Trends Report," Cenzic Report, 2014.
5. S.B. Chavan and B.B. Meshram, "Classification of Web Application Vulnerabilities," IJESIT Volume 2, Issue 2, March 2013.
6. A. Garg and S. Singh, "A Review on Web Application Security Vulnerability," IJARCSSE, volume 3, Issue 1, January, 2013.
7. P. Passeri, "Cyber Attack Timelines," from www.hackmageddon.com, 2021.
8. Johny JHB, Nordin WAFB, Lahapi NMB, Leau YB, "SQL Injection prevention in web application: a review," in Communications in Computer and Information Science, vol. 1487 CCIS, no. January, 2021, pp. 568–585, 2022.
9. Han S, Xie M, Chen HH, Ling Y, "Intrusion detection in cyber-physical systems: techniques and challenges," IEEE Syst J., 2014.
10. Adi, Saltzman, Roi, and Sharabani, "Active Man in the Middle Attacks: A Security Advisory," A whitepaper from IBM Rational Application Security Group, 2009.
11. Xiaowei, Xue, Yuan, and Li, "A survey on web application security," Nashville, TN USA, 2011.
12. D. Kaur, P. Kaur, "Case Study: Secure Web Development, Designing Engineering and Analyzing Reliable and Efficient Software," IGI Global, 2011, pp. 239-250.
13. "Government websites are more vulnerable," article published in weekly Tech Gateway newspaper, Volume 2, Issue 3.
14. R. Barnet, "Web-Hacking-Incident-Database," retrieved from http://projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database on Nov 20, 2015.

Acknowledgment

We would like to express our sincere gratitude to **Dr. Vipin Pal** for their invaluable supervision and guidance throughout this project. Their expertise and continuous support were crucial in shaping the direction of our research.

The insightful feedback and encouragement he provided have greatly enhanced the quality of our work. We are truly grateful for the opportunity to learn under their mentorship and for their dedication to our growth and understanding of web security.

He helped us to lay a strong foundation for its successful completion. Their mentorship has not only helped us overcome challenges but also pushed us to think critically and expand our capabilities in tackling real-world security issues.

Thank You.

Submitted By:

Shobhit (2021UCS1618)

Yash Gautam (2021UCS1690)

Prachi Sah (2021UCS1702)

Supervisor: Dr. Vipin Pal