**Concordia Institute for Information Systems Engineering (CIISE)**
**Concordia University**

INSE 6140 Malware Defenses and Application Security

Project Final Report

**Submitted to:**
Dr. Makan Pourzandi

**Submitted By:**



| Name | Student ID |
|---|---|
| **Devina Shah** | **40238009** |
| **Pratiksha Ashok Kumar Pole** | **40230412** |
| **Zoha Fatima** | **40230911** |

# Emerging Challenges in Web Application Security: Defense Strategies


## Abstract

The exponential growth of web applications is driven by advancements in cloud computing architectures, the prevalence of mobile technology, and the integration of complex functionalities like real-time data processing. This growth reflects an increasing reliance on distributed systems and microservices architecture, however, it escalates security risks, as these applications often process and store sensitive data, making them prime targets for cyberattacks. In the ever-evolving landscape of web application security, emerging challenges necessitate innovative and robust defense mechanisms. The survey essentially presents a comprehensive analysis of advanced defense strategies like an approach to combat DDoS attacks at the application layer, emphasizing the efficacy of firewalls, enhanced CAPTCHA security, honeypots against brute force attacks and SQL injection defense using BURP Suite. By providing a holistic view of the current web application security threats and their countermeasures, it helps guide developers, cybersecurity professionals, and organizations in fortifying their web applications against these challenges.


## Classification of Defense mechanisms

In terms of web application, the attacks can take place on infrastructure and application level. The infrastructure level observes DOS attacks that overwhelm web services with excessive amount of traffic and brute force attacks that target entry points like user accounts and network logins. At the application level, security attacks can be classified into two categories: Active and Passive attacks. Passive attacks involve unauthorized interception, but it does not alter the data or affect the system's resources directly. Web Crawlers are one such that operate covertly, collecting data in a manner that often goes unnoticed by the website owners. Active attacks include Captcha, SQL injection and Cross Site Scripting attacks that directly modify or alter the data stream or the creation of a false stream and can disrupt the normal functioning of web applications. Based on this classification, we have defense mechanisms that encompass strategies, tools and practices aimed at protecting against security threats and attacks. They are essential for protecting sensitive data from unauthorized access and ensuring the integrity, confidentiality, maintain service availability and safeguarding the financial and reputational standing of organizations.

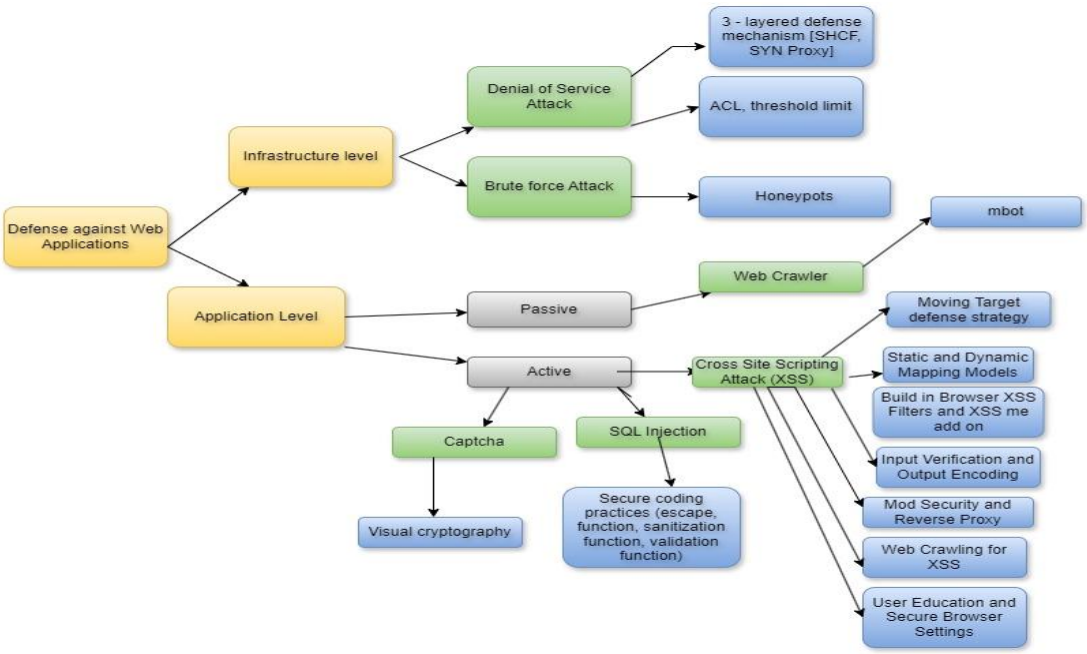


**Fig. Classification of defenses**

## 1. Defense Against Cross Site Scripting Attacks

Cross-site scripting (XSS) is a security vulnerability that occurs when a web application allows users to inject malicious scripts into web pages. The attacker aims to execute malicious scripts in the victim's web browser, often taking advantage of trust that a user has for a particular website.
There are several types of XSS attacks:

- Reflected XSS: This occurs when the malicious script is part of the request to the website, such as in the URL or a form input. The server then includes this script in the response, which is sent back to the user's browser. The browser then executes the script because it appears to be part of the trusted website's content.
- Stored XSS: In this scenario, the malicious script is stored on the server (e.g., in a database, comment, or message board post) and then served to other users who view the affected page. When these users' browsers render the page, the script executes in their browsers.
- DOM-based XSS: This type of XSS involves client-side scripts manipulating the Document Object Model (DOM) in an insecure way. The attack occurs entirely on the client-side, with the malicious payload modifying the DOM environment.

XSS vulnerabilities can have serious consequences such as cookie theft, keylogging, defacement etc.
In this survey seven research papers are gathered to understand the latest research being done to prevent Cross Site Scripting attack.

### i. Moving Target Defense (MTD) Technology

In this mechanism a unique attribute, runtimeId, is introduced to web elements in the application by a system transformer during web page processing. This runtimeId [1] is randomly generated and varies across different instances of a web page. We integrate a detection function to verify runtimeId values before script code execution on the client side. By verifying the runtimeId value on the client browser, we can distinguish between legitimate and injected script code. DOM event handling intercepts and prevents XSS attacks by inspecting attributes and event triggers such as e.g., onclick, onload, onsubmit).

| Web application | page response time ( no protect) | page response time( protect ) | Overhead |
|---|---|---|---|
| DVWA | 12ms | 12.5ms | 4% |
| osCommerce | 268ms | 274ms | 2.3% |
| wordpress | 135ms | 143ms | 5.9% |

**Table 1: The Page Response Time of Non-Protect System and Protected System**

To evaluate the effectiveness of XSSMTD, it was applied on well-known web applications known to have XSS vulnerabilities, including DVWA (Damn Vulnerable Web Application), WordPress, and osCommerce. Experimental results demonstrated that XSSMTD successfully detects and blocks all tested XSS attacks, even when attackers attempt to reuse runtimeId values. It was measured that page response times to assess the overhead of deploying XSSMTD on DVWA, WordPress, and osCommerce show minimal impact on system performance, with an average increase of around 10ms, making XSSMTD practical for real-world deployment scenarios.

### ii. CXSSor: Web Crawling for XSS Vulnerability Detection

CXSSor is an advanced defense mechanism against XSS attacks in web applications, employing proactive techniques such as web crawling to detect and address vulnerabilities effectively. The mechanism functions through a structured process: first, utilizing web crawling to systematically identify potential XSS injection points within target web applications. Next, it tests these points with crafted malicious payloads to simulate XSS attack scenarios and assess the application's response. By analyzing these responses, CXSSor [2] confirms the presence and nature of XSS vulnerabilities. It then generates detailed reports outlining discovered vulnerabilities, aiding developers and security

professionals in promptly addressing and mitigating risks. Overall, CXSSor's comprehensive approach leverages crawler technology and automated testing to provide robust protection against XSS threats.

This proposed mechanism can be used during development of application and to make the application more secure generally. CXSSor offers comprehensive XSS vulnerability coverage through automated web crawling and realistic attack simulation, enabling efficient detection and detailed reporting. However, it may generate false positives/negatives, overlook new attack vectors, and rely heavily on its web crawler's performance. Deployment in live environments could impact web application performance, and continuous updates are essential to address evolving threats. Despite its strengths, CXSSor requires careful management of limitations to ensure effective web application security against XSS attacks.

### iii. ModSecurity and Reverse Proxy in WAF

The paper proposes an effective defense strategy against Cross-Site Scripting (XSS) attacks by integrating ModSecurity, an open-source Web Application Firewall (WAF) module, with a Reverse Proxy within a Web Application Firewall (WAF) environment [3]. ModSecurity filters and monitors HTTP traffic to detect and prevent XSS attacks using predefined rules, including the OWASP Core Rule Set (CRS), while the Reverse Proxy adds a layer of abstraction to enhance control over incoming traffic and conceal server details. The defense mechanism involves traffic filtering, rule based XSS attack detection, blocking of malicious requests, and detailed logging and reporting of detected threats. This integrated approach aims to provide comprehensive protection against XSS attacks by leveraging ModSecurity's capabilities and the security benefits of a Reverse Proxy within a WAF setup.

The inclusion of a reverse proxy enhances security by reducing the attack surface, while real-time monitoring enables swift threat detection and mitigation. Challenges include complex configuration and maintenance of ModSecurity, with potential for false positives or negatives, emphasizing the need for ongoing rule refinement.

### iv. Static and Dynamic Mapping Models

Static and dynamic mapping models are used to detect anomalies and sanitize user inputs effectively. For static web applications (SWA), a mapping model [4] associates each request with a specific SQL query to validate incoming requests against predefined mappings. In dynamic web applications (DWA), query skeletons are generated during a practice session and used to sanitize incoming requests for attack patterns, significantly reducing database space and comparison time. Additionally, each client is assigned a dedicated web service instance for the session, enhancing security by isolating client activities and minimizing the impact of potential attacks.

The defense mechanism efficiently reduces space and time by using query skeletons, enhancing security through client isolation, and covering SQL injection and XSS attacks. However, setup complexity and reliance on known attack patterns may limit its effectiveness against novel or sophisticated attacks, requiring ongoing updates for optimal performance.

### v. Built-in Browser XSS Filters and XSS-Me Add-on

The research focuses on browser-built XSS filters and the XSS-Me add-on for Firefox. The defense mechanisms discussed include Internet Explorer's XSS Filter, which uses regular expressions to modify responses and detect XSS attacks, Google Chrome's XSS Auditor, which scans and blocks JavaScript based on request patterns, and XSS-Me for Firefox [5], an add-on that tests for vulnerabilities by replacing form values with attack strings. The analysis highlights strengths such as comprehensive evaluation and empirical testing against real-world exploits, but also identifies limitations like potential bypasses and the need for manual installation of add-ons.

### vi. Vulnerability analysis of online banking sites to cross-site scripting and request forgery attacks in East Africa

It focuses on vulnerability analysis of online banking in East Africa and emphasizes user education and technical defense measures against XSS and CSRF attacks [6]. Strengths include a focus on user awareness, proactive vulnerability investigation using tools like OWASP's ZAP and XSSer, and efforts to address security threats. Weaknesses include a lack of specific technical defense mechanisms by banks, insufficient user education on security practices, and technical vulnerabilities such as cookie security issues and XSS protection gaps. It highlights the need for a holistic approach combining user education with robust technical defenses to enhance online banking security in the region, requiring collaboration among banks, cybersecurity experts and regulators.

### vii. Input Verification and Output Encoding

The defense strategy proposed in this research focuses on comprehensive measures to mitigate Cross-Site Scripting (XSS) vulnerabilities in an international student website. This includes input verification [7] to validate data based on length, type, syntax, and business rules, output encoding to prevent script execution by encoding all data before outputting, explicit specification of output encoding modes to prevent manipulation by attackers, and awareness of injection points within applications for targeted sanitization and encoding. The approach aims to ensure robust protection against XSS threats by employing a multi-layered defense strategy, combining rigorous input validation, effective output encoding, careful handling of untrusted data, and promoting security-conscious behavior among users.

### viii. Analysis of Cross Site Scripting Defenses

In conclusion, defending against Cross-Site Scripting (XSS) attacks requires a multifaceted approach that combines technical defenses, user education, and continuous updates. Among the various strategies discussed, the best approach involves integrating robust input validation, output encoding, and awareness of injection points within applications. This approach, exemplified by the defense mechanisms like Moving Target Defense (MTD) Technology, CXSSor for web crawling-based detection, and ModSecurity with a Reverse Proxy in a Web Application Firewall (WAF), offers comprehensive protection against XSS vulnerabilities. Challenges include technical complexity, resource-intensive solutions impacting performance, and the need for ongoing updates to address evolving threats. Despite these, the approach strengthens web application security against XSS effectively.

**Cross Compare Different Defense against XSS**

- **Layered Defense Implementation:** Defenses are applied at different levels—client-side (e.g., browser defenses), server-side (MTD, ModSecurity), and network-level (e.g., CXSSor).
- **Automation vs Manual Intervention:** Some methods are highly automated (e.g., CXSSor) for efficient threat detection, while others rely on manual actions or user compliance.
- **Proactive vs Reactive Defense:** Tools like CXSSor are proactive, identifying vulnerabilities before exploitation, whereas browser-based defenses react to attacks as they occur.
- **Coverage vs Specific Focus:** Some techniques provide broad coverage (e.g., input verification, output encoding), while others focus on specific XSS aspects (e.g., static/dynamic mapping models).

## 2. Defense Against SQL Injection Attacks:

SQL injection is a type of attack in which an attacker inserts malicious SQL code into a web form or URL parameter to change the database. This can trick the database into releasing sensitive information or deleting records. The attack works by exploiting flaws in the application's software that

communicates with the database.

Here are the primary types of SQL injection Attacks:

**i. In-band SQL Injection**

- **Error-based SQL Injection:** This type learns about the structure of the database by analyzing its error messages.
- **Union based SQL injection:** The SQL UNION operator combines the results of two or more SLECT statements into a single result, which is returned as part of the HTTP response.

**ii. Blind SQL Injection:**

- **Boolean-based Blind SQL Injection:** This requires sending SQL queries that change the HTTP response based on a TRUE or FALSE query result, allowing attackers to detect whether their injection was successful.
- **Time-based Blind SQL Injection:** In this method, the query causes the database to delay its response, and the response time reveals the outcome of the query.

**iii. Out-of-band SQL Injection:** This occurs when an attacker is unable to launch an attack while collecting information over the same channel. They may rely on the ability to transport data directly from the database server to a system under their control using DNS or HTTP protocols, depending on the database server's capabilities.

**Effective Defense Mechanism Against SQL Injection**

After analyzing six research papers on defense mechanisms, three strategies were consistently identified as effective against SQL Injection (SQLi) attacks. These approaches include sanitization, escape, and validation. Each method contributes significantly to a multilayered defense.

i. **Using 'mysql-real-escape-string()' for escaping Special Characters:** The mysql_real_escape_string() function is essential for protecting SQL queries.[12][13][8]. It hides potentially harmful characters entered by users, such as single quotes, double quotes, backslashes, and NULL bytes. This function prevents SQL statements from being terminated or modified by prefixing certain special characters with backslashes. This is especially critical for inputs that will be used in SQL commands, as it ensures that the data does not distort or change the intended SQL execution logic.

ii. **Applying stripslashes() to Remove Unwanted Backslashes:** While mysql_real_escape_string() inserts the necessary backslashes to safeguard SQL queries, stripslashes() eliminates any extra backslashes.[11][12]. This ensures that the data processed by the application is clean and clear of misleading characters that could be exploited in SQLi, which is especially important when inputs are pre-escaped due to other settings.

iii. **Leveraging is_numeric() to Validate Numeric Input:** The is_numeric() method is critical for ensuring that data intended to be numeric is free of any alphabetic or symbolic intrusions that could cause SQLi.[8][12][11][9]. For example, if the user input is expected to be an ID (which is typically numeric), this function validates its numeric nature before the application runs it in a SQL query. This validation protects against a common sort of SQL injection, in which attackers change IDs with malicious SQL code.

| Defense Technique | Pros | Cons |
|---|---|---|
| Escaping (‘mysql_real_escape_string()’) | • Straightforward implementation in any SQL based application.<br>• Prevents attackers from breaking out of data context.<br>• Quick fix for legacy systems. | • Provides a false sense of security, not foolproof against nested queries.<br>• Dependent on proper usage within the code.<br>• Potentially obsolete with newer database technologies. |
| Sanitization (‘stripslashes()’) | • Removes problematic characters that could alter SQL commands.<br>• Reduces risk of accidentally stored code execution. | • May mistakenly change legitimate data entries.<br>• Requires careful implementation to avoid creating additional bugs. |
| Validation (‘is_numeric()’) | • Ensures that data conforms to expected types, avoiding type mismatches.<br>• Simple to apply to data fields that require numeric input.<br>• Can be combined with other security measures for enhanced protection. | • Limited to data type validation, not a comprehensive solution.<br>• Ineffective against complex injections not involving data types.<br>• Needs to be tailored for each specific input scenario. |

The integration of these services into online web-applications, as detailed in the evaluated research publications, provides a strong framework for preventing SQL injections. By escaping inputs, cleaning data, and validating forms, developers can greatly reduce the risk of SQL injection, hence improving the security and integrity of their database interactions.

## 3. Three Layer Defense Mechanism against DDoS attack

Distributed Denial of Service (DDoS) attacks are malicious attempts to disrupt normal web services by overwhelming them with an excessive amount of traffic which result in significant operational disruptions and revenue losses. Given the severity of these attacks, a three-layer defense mechanism is designed to ensure the continued availability of web services by employing a combination of statistical filtering and traffic limiting strategies across three distinct layers of network infrastructure:

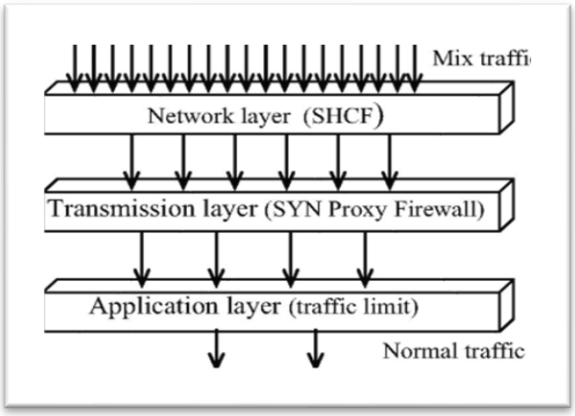


**Fig. Three-Layered Defense Mechanism**

- **Defending at network layer:**

The network layer is responsible for packet forwarding and each IP packet has the source, destination IP addresses and Time-to-Live (TTL) value. To filter out illegitimate traffic, which is spoofed IP packets, it utilizes Simplified Hop Count Filtering (SHCF) [14] and block packets with inconsistent hop counts relative to their claimed source addresses.

The TTL is an 8-bit field in the IP header and is decremented by one by each router that forwards the packet. When TTL reaches zero, the packet is discarded. This prevents packets from looping indefinitely in the network. First, the hop count is calculated by subtracting the final TTL (when the packet arrives at the destination) from the initial TTL value. The system constructs an SHCF table by grouping IP addresses based on the first 24 bits of their address prefixes.

The computed hop count is compared with the hop count from the SHCF table for the corresponding 24-bit address prefix. If the difference between the computed hop count and the stored hop count is more than two, the packet is likely spoofed and is dropped

- **Defending at transport layer:**

A typical TCP connection starts with a three-way handshake: the client sends a SYN (synchronize) packet to initiate a connection, the server responds with a SYN-ACK (synchronize-acknowledge) packet, and the client sends an ACK (acknowledge) packet to establish the connection.

In a SYN flood attack, an attacker sends a high volume of SYN packets to a server but does not respond with the final ACK. The server allocates resources for each connection which gets exhausted leading to denial of service for legitimate requests.

The SYN Proxy Firewall acts as an intermediary between the client and the server. When a SYN packet is received, the firewall intercepts it and responds to the client on behalf of the server by sending a SYN-ACK packet. The client will respond with an ACK and the firewall understands that it's a legitimate request. If the connection was coming from attacker, the final ACK from the client will never arrive, so the firewall will terminate the connection which prevents the allocation of server resources for this bogus request.

- **Defending at application layer:**

Sometimes Attackers can use genuine IP addresses to send HTTP requests to consume server resources like bandwidth. The system implements a mechanism to equally distribute available bandwidth among all clients and the idea is to treat all traffic equally whether it's from a legitimate user or an attacker. It does so by setting a limit (quota) on the amount of traffic each client can send. This quota is based on typical user behavior observed under normal conditions. When a client exceeds this quota, The system then reduces the bandwidth allocated - only be allowed a fraction (like 1/10th) of their normal bandwidth allocation.

### Defenses on UDP DDoS Flood Cyber Attack

In a UDP flood, the system receives many UDP packets, which can overwhelm the system's resources, leading to slow response times or a complete crash. The two defenses that could be employed against these attacks are:
Access Control Lists (ACLs): Blocks all private IP addresses by defining rules that specifically deny traffic from these addresses. Manages traffic flow by checking incoming and outgoing packets against these rules. If a packet's source IP address matches the range of private IP addresses defined in the ACL rule, the packet is blocked or dropped.
Threshold Limit: Helps manage network resources by setting a cap on the rate of traffic- up to 10000 packets per second [15], ensuring that system resources like CPU and bandwidth are used efficiently. This prevents network services from being overloaded by excessive traffic.

| Performance Metric | Before Defenses | After Defenses (Selected Results) |
|---|---|---|
| TCP Throughput | 94 Mbps (normal) | 94 Mbps (Threshold Limit) |
| | 0.36 Mbps (during attack) | 53.37 Mbps (ACLs) |
| Round-Trip Time (RTT) | 0.62ms (normal) | 25.98ms (Threshold Limit) |
| | 26.42ms (during attack) | 26.007ms (ACLs) |
| CPU Utilization | 0.3% - 0.5% (normal) | 3% (Threshold Limit) |
| | 24.9% (during attack) | 15% (ACLs) |

**Fig. Comparative analysis of the performance of a web server running Linux Ubuntu 13, before and after applying defence mechanisms against a UDP flood attack.**

| PROS | CONS |
|---|---|
| **High Efficiency in Filtering Illegitimate Traffic:** The combination of statistical filtering and traffic separating illegitimate from legitimate traffic. | **Potential for Legitimate Traffic Delay or Blockage:** delay or block legitimate traffic when attack patterns mimic normal traffic, leading to service disruptions. |
| **Improved Traffic Management:** ACLs can block known malicious sources, while the Threshold Limit can control the rate of traffic, reducing server overloads. | **Potential for Legitimate Traffic Blockage:** especially if the ACLs are too restrictive or the threshold is set too low. |

## 4. Honeypot Against Brute Force Attacks

Brute force targets various entry points like user accounts, network logins and encrypted files.

Safeguarding data and protecting access through the network on the data such as using Honeypots. Honeypots create a shadow server by tricking attackers into believing they have gained access to a real server. It stores the attacker's IP, methods, and behaviors.

- **The Brute Force Attack Simulation -** the Medusa Tool [16] is used that identifies the target system (SSH, FTP, HTTP) and its IP address. It does this by compiling a dictionary list of usernames and passwords.

The outcome of using honeypots is that attackers are diverted to the shadow server, thereby protecting the real server from unauthorized access. Attempts to create files, crack systems, or install backdoors by the attacker on the shadow server is also not possible.

- **Log Analysis with Kippo-** data about attacker's actions on the shadow server is being logged by a tool called Kippo where system administrators could monitor and analyze attack patterns.

| PROS | CONS |
|---|---|
| **Diversion:** mislead attackers; protects real servers from unauthorized access. | **Resource Intensive:** Requires significant resources for setup and maintenance. |
| **Attacker Profiling:** gathers comprehensive data on attacker methods and behavior. | **Potential for Misdirection:** May fail to attract skilled attackers, leading to ineffective trapping. |

## 5. Visual Cryptography against text-based captcha attack

Text based Captcha's are attacked primarily through two methods: binarization, which involves converting the colorful captcha images to black and white for easier character separation, and vertical segmentation that divides the CAPTCHA into individual characters for subsequent recognition and decoding by automated system.

To better defend text-based captcha attacks, online banking systems use Visual Cryptography that is implemented by dividing a secret or image into two shares [17]. During registration, one share is provided to the user, while the other is retained by the bank. When the user logs in, they submit their share alongside their credentials. The bank then combines this share with its own to recreate the original image. If the combined image matches the predetermined one, it verifies the authenticity of the website.

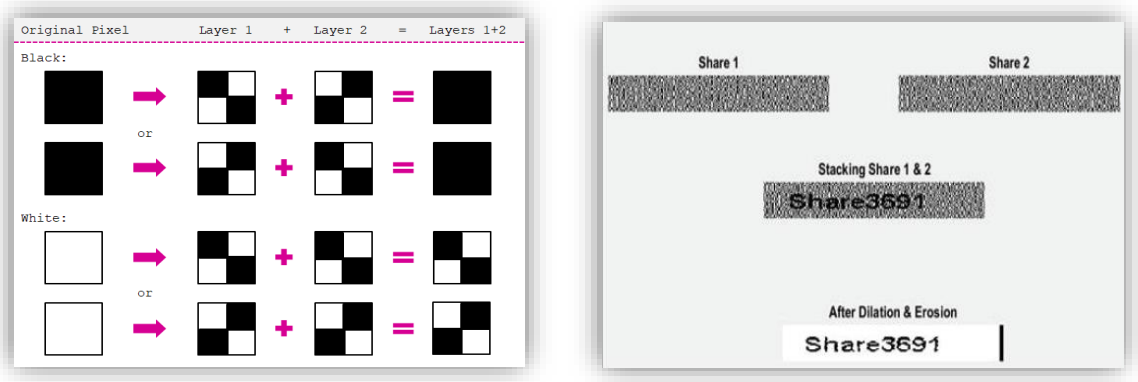


**Fig. Visual Cryptography**

| PROS | CONS |
|---|---|
| **Advanced Security Layer:** Offers a sophisticated layer of security against phishing, surpassing traditional CAPTCHA systems. | **Computational Load:** Increased computational requirements for both customers and banks. |
| **Dual-Share Validation:** Enhances authentication reliability through a unique dual-share visual cryptography method. | **Frequent Updates:** Requires regular updates of secrets, leading to administrative challenges and user inconvenience. |

### 6. Mbots against web crawlers

To protect web applications from unauthorized or non-cooperative web crawlers, we use robots.txt to specify which parts of the website crawlers are allowed or disallowed from accessing. While compliant crawlers will adhere to these rules, this method won't stop non-cooperative crawlers.

**Mbot** is designed to mimic the actions of crawlers that do not follow the guidelines set in the robots.txt files, it collects data on how the site responds to its actions.
The reaction of websites to Mbot's [18] activities provides insight into their defense strategies, such as the implementation of automated content access protocols, CAPTCHAs, and real-time bot detection.

| PROS | CONS |
|---|---|
| **Enhanced Security Measures**: identifying weak spots that non-cooperative crawlers could exploit. | **False Positives**: could potentially trigger security measures leading to false positives in system logs. |
| **Realistic Threat Simulation**: provide a real-world testing environment by simulating actual attack patterns to improve the robustness of security systems. | **Complex Programming**: Designing and programming mbots can be highly complex as requires advanced skills and resources. |

## Analysis and Cross Comparison of Classifications

In this section we analyze the different defense mechanisms discussed and summarize it in a table as shown below. We observe that there are many defenses that can be used to prevent or detect multiple attacks.

For example –

- We can see that we can use Captcha to prevent Captcha based attacks or Web Crawlers based attack as any bots, be it good or bad crawlers, they are automated bots, so they get filtered out in the captcha process during authentication.
- Similarly multiple defense processes which focus on the user input on an application can be used to prevent both SQL Injection and Cross site scripting as both use forms, login fields or feedback textbox as one of the primary starting points of the attack.

Even though there are situations where a defense method can provide security against a lot of vulnerabilities, we cannot conclude one defense as the most secure one. Each has its own plus points and limitations. Some are a little challenging to implement and maintain, while others may have a huge financial overhead or scalability issue.

The kind of defense that we should employ for a particular application also depends on the risk evaluation, ease of use and the purpose of the application. A banking application may employ different defenses as compared to an ecommerce website or social media website while all are susceptible to all kinds of attack. It highly depends on the kind of data they store and the services they provide.

| Attacks / Defences | DOS | XSS | Brute Force | Captcha | SQL | Web Crawlers |
|---|---|---|---|---|---|---|
| 3 layered Defense | Y | N | N | N | N | N |
| ACL & threshold limit | Y | N | N | N | N | N |
| Text-based captcha | N | N | N | Y | N | Y |
| Visual Cryptography | N | N | N | Y | N | N |
| Honeypots | N | N | Y | N | N | Y |
| Mbots | N | N | N | Y | N | Y |
| Static and Dynamic Mapping | N | Y | N | N | N | N |
| XSS-Me Add-on | N | Y | N | N | N | N |
| MTD Technology | N | Y | N | N | N | N |
| Input Verification & Output Encoding | N | Y | N | N | N | N |
| ModSecurity and Reverse Proxy in WAF | N | Y | N | N | N | N |
| CXSSor: Web Crawling | N | Y | N | N | N | N |
| Input sanitization | N | Y | N | N | Y | N |
| Validation | N | Y | N | N | Y | N |
| Escape | N | Y | N | N | Y | N |

**Table: Cross comparison between classification**

## Conclusion

An overview of our studied defenses shows that if we consider the example of a banking web application, when a user tries to login, a username and password captcha is prompted. brute forcing is possible on the login page at the web browser level. Honeypots/shadow server at the web server level prevents brute force attacks. Distorted and overlapping characters in captcha prevents text-based captcha attacks. Visual crypto acts as an extra layer of protection in banking application where user has one share of the image and the bank db has the other share of image. The two shares combined gives the decoded image at the web server. Mbots are used in analyzing the attack patterns of non-cooperative crawlers. A three-layered defense mech is implemented to protect ddos attack by filtering the traffic. Moving target defense against xss attack- the verification of runtimeid and html script ensures that injected scripts aren't executed.

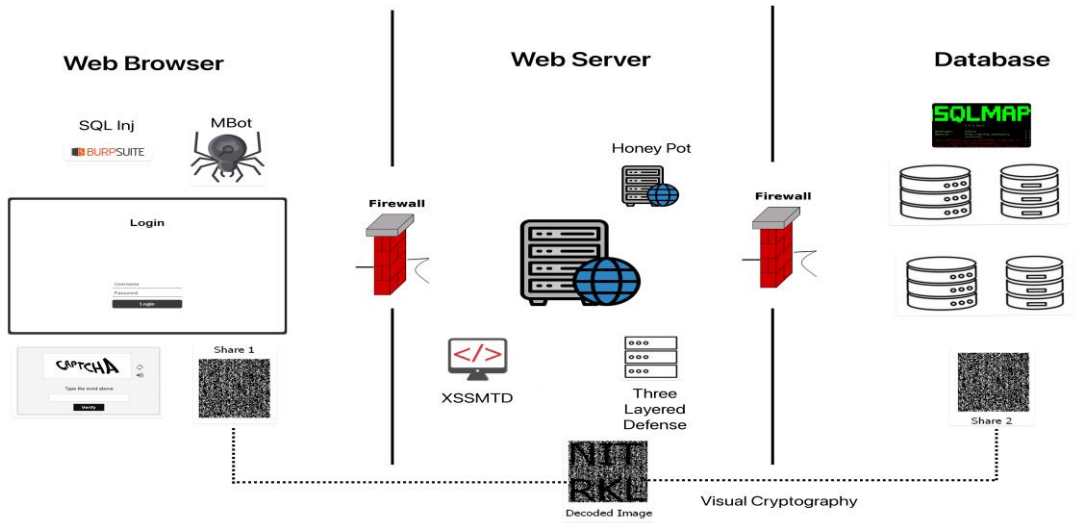


**Fig. Overview of Defense Mechanisms**

## Team Member Contribution

| Team Member | Research Area | Key Contributions |
|---|---|---|
| Devina Shah | Defense Against Cross-Site Scripting (XSS) | Analyzed contemporary XSS defense methods and integrated findings into web app security practices. Worked on classification of various defence mechanisms to structure the flow of paper. |
| Pratiksha Pole | Defense Against Denial of Service (DOS) Attacks, Brute force, Captcha attacks, Web Crawlers | Investigated robust techniques to mitigate DOS attacks, Brute force and Captcha attacks thereby enhancing network resilience. Worked on structure and overview of the defense strategies. |
| Zoha Fatima | Defense Against SQL Injection (SQLi) Attacks | Explored SQLi prevention strategies, contributing to database security enhancements. Worked on weighing out pros and cons within each class of research. |

## References


[1] P. Chen, H. Yu, M. Zhao and J. Wang, "Research and Implementation of Cross-site Scripting Defense Method Based on Moving Target Defense Technology," *2018 5th International Conference on Systems and Informatics (ICSAI)*, Nanjing, China, 2018, pp. 818-822, doi: 10.1109/ICSAI.2018.8599463.

[2] H. Guan, D. Li, H. Li and M. Zhao, "A Crawler-Based Vulnerability Detection Method for Cross-Site Scripting Attacks," *2022 IEEE 22nd International Conference on Software Quality, Reliability, and Security Companion (QRS-C)*, Guangzhou, China, 2022, pp. 651-655, doi: 10.1109/QRS-C57518.2022.00103.

[3] R. A. Muzaki, O. C. Briliyant, M. A. Hasditama and H. Ritchi, "Improving Security of Web-Based Application Using ModSecurity and Reverse Proxy in Web Application Firewall," *2020 International Workshop on Big Data and Information Security (IWBIS)*, Depok, Indonesia, 2020, pp. 85-90, doi: 10.1109/IWBIS50925.2020.9255601.

[4] P. A. Sonewar and S. D. Thosar, "Detection of SQL injection and XSS attacks in three tier web applications," *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*, Pune, India, 2016, pp. 1-4, doi: 10.1109/ICCUBEA.2016.7860069.

[5] B. Mewara, S. Bairwa and J. Gajrani, "Browser's defenses against reflected cross-site scripting attacks," *2014 International Conference on Signal Propagation and Computer Technology (ICSPCT 2014)*, Ajmer, India, 2014, pp. 662-667, doi: 10.1109/ICSPCT.2014.6884928.

[6] G. Buah, S. Memusi, J. Munyi, T. Brown and R. A. Sowah, "Vulnerability Analysis of Online Banking Sites to Cross-Site Scripting and Request Forgery Attacks: A Case Study in East Africa," *2021 IEEE 8th International Conference on Adaptive Science and Technology (ICAST)*, Accra, Ghana, 2021, pp. 1-5

[7] T. Wang, D. Zhao and J. Qi, "Research on Cross-site Scripting Vulnerability of XSS Based on International Student Website," *2022 International Conference on Computer Network, Electronic and Automation (ICCNEA)*, Xi'an, China, 2022, pp. 154-158, doi: 10.1109/ICCNEA57056.2022.00043.

[8] G. P. Bherde and M. A. Pund, "Recent attack prevention techniques in web service applications," *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*, Pune, India, 2016, pp. 1174-1180, doi: 10.1109/ICACDOT.2016.7877771.

[9] I. Tasevski and K. Jakimoski, "Overview of SQL Injection Defense Mechanisms," *2020 28th Telecommunications Forum (TELFOR)*, Belgrade, Serbia, 2020, pp. 1-4, doi: 10.1109/TELFOR51502.2020.9306676.

[10] J. Fonseca, M. Vieira and H. Madeira, "Evaluation of Web Security Mechanisms Using Vulnerability & Attack Injection," in *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 5, pp. 440-453, Sept.-Oct. 2014, doi: 10.1109/TDSC.2013.45.

[11] D. Mitropoulos, P. Louridas, M. Polychronakis and A. D. Keromytis, "Defending Against Web Application Attacks: Approaches, Challenges and Implications," in *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 2, pp. 188-203, 1 March-April 2019, doi: 10.1109/TDSC.2017.2665620.

[12] R. Bouafia, H. Benbrahim and A. Amine, "Automatic Protection of Web Applications Against SQL Injections: An Approach Based On Acunetix, Burp Suite and SQLMAP," *2023 9th International*

*Conference on Optimization and Applications (ICOA)*, AbuDhabi, United Arab Emirates, 2023, pp. 1-6, doi: 10.1109/ICOA58279.2023.10308827.

[13] A. Tekerek, C. Gemci and O. F. Bay, "Development of a hybrid web application firewall to prevent web-based attacks," *2014 IEEE 8th International Conference on Application of Information and Communication Technologies (AICT)*, Astana, Kazakhstan, 2014, pp. 1-4, doi: 10.1109/ICAICT.2014.7035910.

[14] Z. Wu and Z. Chen, "A Three-Layer Defense Mechanism Based on WEB Servers Against Distributed Denial of Service Attacks," *2006 First International Conference on Communications and Networking in China*, Beijing, China, 2006, pp. 1-5, doi: 10.1109/CHINACOM.2006.344851.

[15] S. S. Kolahi, K. Treseangrat and B. Sarrafpour, "Analysis of UDP DDoS flood cyber-attack and defense mechanisms on Web Server with Linux Ubuntu 13," *2015 International Conference on Communications, Signal Processing, and their Applications (ICCSPA'15)*, Sharjah, United Arab Emirates, 2015, pp. 1-5, doi: 10.1109/ICCSPA.2015.7081286.

[16] A. Nursetyo, D. R. Ignatius Moses Setiadi, E. H. Rachmawanto and C. A. Sari, "Website and Network Security Techniques against Brute Force Attacks using Honeypot," *2019 Fourth International Conference on Informatics and Computing (ICIC)*, Semarang, Indonesia, 2019, pp. 1-6, doi: 10.1109/ICIC47613.2019.8985686.

[17] M. A. Snober, A. Droos and Q. A. Al-Haija, "Prevention of phishing website attacks in online banking systems using visual cryptography," *6th Smart Cities Symposium (SCS 2022)*, Hybrid Conference, Bahrain, 2022, pp. 168-173, doi: 10.1049/icp.2023.0391.

[18] R. Dev Chandna, P. Chaubey and S. C. Gupta, "Defense response of search engine websites to non-cooperating crawlers," *2012 World Congress on Information and Communication Technologies*, Trivandrum, India, 2012, pp. 219-223, doi: 10.1109/WICT.2012.6409078.

[19] A. Praseed and P. S. Thilagam, "DDoS Attacks at the Application Layer: Challenges and Research Perspectives for Safeguarding Web Applications," in *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 661-685, Firstquarter 2019, doi: 10.1109/COMST.2018.2870658.

[20] X. Ling-Zi and Z. Yi-Chun, "A Case Study of Text-Based CAPTCHA Attacks," *2012 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, Sanya, China, 2012, pp. 121-124, doi: 10.1109/CyberC.2012.28.