SECURITY TESTING ON FILE UPLOAD WEB APPLICATIONS BASED ON THE YII 2 FRAMEWORK

Rico Setyawan[1)], Rheva Anindya Wijayanti[2,*)], Hermawan Setiawan[3)]

1) Mahkamah Konstitusi Republik Indonesia, rico.setyawan@mkri.id

2) Badan Siber dan Sandi Negara, rhevaanindyaw@gmail.com

3) Program Studi Rekayasa Kriptografi, Politeknik Siber dan Sandi Negara, hermawan.setiawan@poltekssn.ac.id

Article History

Sent: 2 Sep 2025
Received: 21 May 2026
Published: 22 may 2026

Keywords:

Security Testing
File Upload
Yii 2 Framework
STRIDE
DREAD

Abstrak

Kerangka kerja Yii 2 banyak digunakan untuk mengembangkan aplikasi web modern karena kemampuan performa tinggi dan arsitektur yang terstruktur. Namun, kerentanan nyata seperti CVE-2018-7269 menunjukkan bahwa masalah keamanan masih dapat terjadi meskipun terdapat mekanisme bawaan. Penelitian ini mengevaluasi ketahanan keamanan aplikasi unggah berkas berbasis Yii 2 dengan menggunakan kerangka pemodelan ancaman STRIDE dan model penilaian risiko DREAD. Empat vektor serangan—SQL Injection (SQLi), Cross-site Scripting (XSS), Remote Access Trojan (RAT) melalui unggahan berkas, dan Buffer Overflow—disimulasikan dalam lingkungan terkontrol dengan mengacu pada CVE dunia nyata. Hasil eksperimen menunjukkan bahwa aplikasi berhasil memblokir seluruh serangan melalui validasi bawaan, pembatasan masukan, dan pemfilteran ketat tipe MIME. Penilaian DREAD mengungkapkan tingkat risiko tinggi untuk SQLi (7.6) dan RAT (7.8), sedangkan XSS (5.6) dan Buffer Overflow (6.2) dikategorikan pada risiko sedang. Temuan ini menunjukkan bahwa aplikasi berbasis Yii 2 yang diuji memiliki mekanisme keamanan yang efektif dalam kondisi pengujian, sekaligus menekankan pentingnya pengujian berkelanjutan dan penerapan SSDLC.

Abstract

The Yii 2 framework is widely used for developing modern web applications due to its high-performance capabilities and structured architecture. Nevertheless, real-world vulnerabilities, such as CVE-2018-7269, highlight persistent security concerns despite the built-in mechanisms. This study evaluates the security resilience of a Yii 2-based file upload application using the STRIDE threat modelling framework and the DREAD risk assessment model. Four attack vectors—SQL Injection (SQLi), Cross-site Scripting (XSS), Remote Access Trojan (RAT) via file upload, and Buffer Overflow—were simulated in a controlled environment referencing real-world CVEs. Experimental results show that the application successfully blocked all vectors through strict MIME-type filtering, input restriction, and built-in validation. DREAD scoring revealed high-risk severity for SQLi (7.6) and Remote Attacks (7.8), while XSS (5.6) and Buffer Overflow (6.2) were categorised as medium risks. These findings indicate that the tested Yii 2-based application demonstrates effective security mechanisms under the evaluated conditions, while highlighting the importance of continuous testing and SSDLC integration.

1. INTRODUCTION

The rapid growth of information technology has transformed how organisations deliver services, particularly through web applications. These applications are widely used for communication, transactions, and the sharing of information. However, their popularity also makes them attractive targets for attackers. Common attack vectors include SQL Injection, Cross-Site Scripting (XSS), Buffer Overflow, and malicious file uploads, all of which can compromise system integrity and user trust [5].

The file upload functionality, in particular, poses significant security risks, as improperly validated files may allow for remote code execution or data breaches [1][3][4]. While numerous frameworks offer security features, their actual resilience against real-world attacks often remains untested in practice [6].

Numerous studies have evaluated the security of web application frameworks and have demonstrated that frameworks can help reduce common vulnerabilities through built-in protections and standardized development practices. However, most of these studies focus on general security features or conceptual comparisons, without conducting empirical attack-based validation on specific functionalities such as file upload mechanisms. Consequently, the actual effectiveness of these frameworks against real-world exploit scenarios remains insufficiently explored [6].

The Yii 2 framework is a modern PHP framework that incorporates built-in mechanisms, including input validation, authentication, access control, and file handling restrictions [2][9]. Despite these features, empirical evaluations of Yii 2's effectiveness against file-upload-related exploits are limited. Existing studies typically discuss Yii 2 in terms of performance and usability, rather than validating its resistance to concrete security threats.

This study addresses that gap by systematically testing a Yii 2-based file upload application against four representative attacks—SQL Injection, Cross-Site Scripting, Remote Access Trojan (via file upload), and Buffer Overflow. Using the STRIDE threat modeling framework and DREAD risk assessment [7], the study not only classifies threats but also quantifies their severity. The findings provide evidence-based insights into the robustness of Yii 2's built-in security mechanisms and highlight the importance of integrating structured threat modeling and risk analysis into the Secure Software Development Lifecycle (SSDLC) [8].

2. BASIC CONCEPT

Information Security. Information security is the practice of protecting data from unauthorised access, alteration, or destruction. These principles provide the foundation for designing and evaluating secure web applications. The CIA Triad—confidentiality, integrity, and availability—commonly guides this process [8].

Secure Software Development Lifecycle (SSDLC). SSDLC integrates security into every phase of software development, emphasising proactive identification of vulnerabilities through practices such as threat modelling, code review, and penetration testing. Its three pillars—Knowledge, Touchpoints, and Risk Management—serve as a framework for developing applications that remain resilient against evolving threats [8].

Yii 2 Framework and Security Features. Yii 2 is a modern, component-based PHP framework built on the MVC (Model–View–Controller) architecture. It includes built-in features for input validation, authentication, access control, and role-based access management, which strengthen application security. Yii 2's extensibility also enables the integration of advanced security measures, making it suitable for applications that require both scalability and robustness [9].

Security Testing Approach. Security testing is the process of identifying vulnerabilities, threats, and risks within software systems to prevent exploitation and ensure operational reliability and integrity. In this study, testing is structured around STRIDE threat modelling and DREAD risk assessment, which together provide a systematic approach to threat identification and severity quantification [7]. This combination ensures that testing is not only theoretical but also directly aligned with practical attack vectors.

3. RELATED WORK AND STATE OF THE ART ANALYSIS

Recent studies have extensively explored web application security, particularly in the context of modern frameworks. Several researchers have demonstrated that frameworks such as Laravel, Django, and Spring provide built-in mechanisms to mitigate common vulnerabilities, including SQL Injection and Cross-Site Scripting (XSS). These studies highlight that the use of frameworks can significantly reduce developer errors by enforcing secure coding practices.

However, most existing research primarily focuses on general security features or comparative performance evaluations of frameworks rather than conducting empirical validation through real attack simulations. For instance, prior works often evaluate security based on configuration analysis or theoretical discussions, without testing how frameworks behave under actual exploit scenarios.

In the context of PHP frameworks, Yii 2 has been discussed in terms of its performance, scalability, and modular architecture. While some studies mention its built-in security features—such as input validation, role-based access control, and query parameterisation—there is limited empirical evidence demonstrating its effectiveness against specific attack vectors, particularly those related to file upload mechanisms.

Furthermore, research on file upload vulnerabilities has shown that this functionality remains one of the most exploited entry points in web applications, often leading to Remote Code Execution (RCE) or malware injection. Despite this, few studies integrate structured threat modelling (such as STRIDE) and quantitative risk assessment models (such as DREAD) to evaluate framework-level security.

Therefore, this study distinguishes itself by:

1. Performing attack-based empirical testing rather than theoretical analysis
2. Focusing specifically on file upload vulnerabilities
3. Applying a combined STRIDE–DREAD approach for both threat identification and risk quantification
4. Evaluating the real-world resilience of Yii 2 against multiple attack vectors

This positions the research within the current state-of-the-art while addressing gaps not sufficiently explored in previous studies.

4. METHODOLOGY

The application was developed with basic functionalities, including user login and file upload. The security analysis was conducted using the STRIDE threat modelling framework and the DREAD risk assessment model.

Research Environment. This research was conducted in a controlled local environment, which includes both hardware and software components as listed below:

Table 1. Virtual Environment

Hardware	
Device	Lenovo B490 Laptop
RAM	4 GB
Storage	500 GB HDD
Software	
Operating System	Windows 10 (64-bit)
Web Server	Apache PHP Server
Database Server	MySQL
Framework	Yii 2 (PHP Framework)

Knowledge. In this study, knowledge is derived from Common Vulnerabilities and Exposures (CVEs) associated with the Yii 2 framework, which serve as references for simulating real-world web

attacks. The following table lists selected CVEs that represent common threats affecting Yii 2-based web applications:

Table 2. CVE about attacks on Yii 2

No	CVE ID	Attack Type
1	CVE-2018-7269	SQL Injection
2	CVE-2015-3397	Cross-site Scripting (XSS)
3	CVE-2018-8073	Remote Attack
4	CVE-2019-11036	Buffer Overflow

Touchpoints and Risk Management.

1. STRIDE-Based Threat Modelling

   The first stage of building a secure system is identifying and understanding potential threats that may occur in a web-based file uploader application built using the Yii 2 PHP framework.

   a. Spoofing: Occurs when an attacker successfully impersonates a legitimate user within the system.
   b. Tampering: Involves unauthorised modification, deletion, or insertion of data.
   c. Repudiation: Happens when a user denies acting, and the system lacks sufficient evidence to prove otherwise.
   d. Information Disclosure: Refers to the exposure of sensitive information to unauthorised users.
   e. Denial of Service (DoS): Entails disabling or reducing the availability of services to legitimate users.
   f. Elevation of Privilege: Arises when a user gains unauthorised, higher-level access to system resources.

   The STRIDE model was applied to simulate and evaluate four representative attack vectors based on historical vulnerabilities (as referenced from CVE reports): SQL Injection, Cross-site Scripting (XSS), Remote Attack, and Buffer Overflow.

2. DREAD-Based Risk Assessment

   To evaluate risk severity for each identified threat, the study implements the DREAD Security Assessment Model, which quantifies risk across five dimensions:

   a. Damage Potential (D) – The extent of harm if the threat is successfully exploited.
   b. Reproducibility (R) – The ease with which an attack can be repeated.
   c. Exploitability (E) – How simple it is to exploit the vulnerability.
   d. Affected Users (A) – The number of users impacted by the threat.
   e. Discoverability (D) – The ease of discovering the vulnerability.

   Each category is rated on a scale from 0 to 10, and the average score is used to determine the overall risk level (e.g., *Medium* or *High*). The results are shown below:

Table 3. Table of linguistic variables and their ranges

No.	Linguistic Variable	Very Low / Probably / Negligible	Low / Slight / Likelihood	Somewhat Low / Moderate / Satisfiable	Medium / Almost / Critical	High / Catastrophic / Vital
1	Damage Potential (D)	0 – 2	1 – 4	3 – 6	5 – 8	7 – 10
2	Reproducibility (R)	0 – 2.5	1.5 – 4	3.5 – 5.6	5.5 – 8	7.5 – 10
3	Exploitability (E)	0 – 3	2 – 5	4 – 7	6 – 9	8 – 10
4	Affected Users (U)	0 – 2	1 – 4	3 – 6	5 – 8	7 – 10
5	Discoverability (D)	0 – 2	1.5 – 5	3.5 – 7	5.5 – 9	7.5 – 10
6	Fuzzy Risk Level	0 – 10 *(Very Low)*	7 – 17 *(Low)*	14 – 24 *(Somewhat Low)*	21 – 31 *(Medium)*	28 – 37 *(Somewhat High)*35 – 43 *(High)*40 – 50 *(Very High)*

5. RESULT AND DISCUSSION

1. STRIDE Threat Model Analysis

The STRIDE model classifies threats into six categories: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. The four attack vectors analyzed in this study—SQL Injection, Cross-site Scripting (XSS), Remote Attack via file upload, and Buffer Overflow—were mapped as follows:

SQL Injection

- Tampering (T): SQL queries can be modified to alter database content or structure.
- Information Disclosure (I): Sensitive data can be exposed through manipulated queries.
- Elevation of Privilege (E): Attackers may obtain administrative privileges. → Classified as *High risk* due to direct impact on data integrity and confidentiality.

Cross-site Scripting (XSS)

- Spoofing (S): Attackers may impersonate legitimate users via session hijacking.
- Information Disclosure (I): User credentials, cookies, or personal data can be stolen. → Categorized as *Medium risk*, as the impact is primarily on end-users rather than the entire system.

Remote Attack (RAT/File Upload)

- Tampering (T): Malicious files can be uploaded to the server.
- Elevation of Privilege (E): Malware execution allows complete system takeover. → Considered *High risk* because it enables complete attacker control over the system.

Buffer Overflow

- Denial of Service (D): The application may crash due to excessive input.
- Elevation of Privilege (E): Exploitation can lead to arbitrary code execution. → Categorized as *Medium risk* since successful exploitation often depends on specific runtime conditions.

The classification of the four selected attacks, based on STRIDE and their corresponding quantified risk levels, is presented in Table 4. SQL Injection and Remote Attack fall under the *high-risk category, while Cross-site Scripting (XSS) and Buffer Overflow are classified as Medium-Risk*.

Table 4. STRIDE Threat Assessment

Threat	S	T	R	I	D	E	Avg. DREAD Score	Risk Level (Linguistic)
SQL Injection	✓				✓	✓	7.6	High (Catastrophic)
Cross-site Scripting	✓				✓		5.4	Medium (Critical)
Remote Attack		✓				✓	7.8	High (Catastrophic)
Buffer Overflow					✓	✓	5.1	Medium (Critical)

SQL Injection

A series of eight SQL injection payloads was executed against the login form (Table 5). All attempts failed, confirming that Yii 2 effectively neutralises SQLi vectors.

Table 5. SQL Injection Payloads and Application Responses

No.	Username Input	Password Input	Result
1	admin' --	admin' --	Failed
2	admin' #	admin' #	Failed
3	admin'/*	admin'/*	Failed
4	' or 1=1--	' or 1=1--	Failed
5	' or 1=1#	' or 1=1#	Failed
6	' or 1=1/*	' or 1=1/*	Failed
7	') or '1'='1--	') or '1'='1--	Failed
8	') or '1'='1--	') or '1'='1--	Failed

This resilience can be attributed to Yii 2's ORM Query Builder and ActiveRecord mechanisms, which automatically parameterise user inputs and prevent raw string concatenation in SQL statements. Additionally, Yii 2 applies built-in validation rules to form inputs, ensuring that only properly structured data is processed. These protections are enabled by default and require no special configuration.

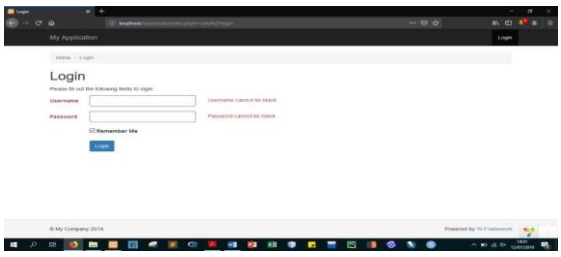


Figure.1 Login Form Validation (Yii 2)
All login form fields must be filled. If left empty, the system issues a warning before allowing submission.

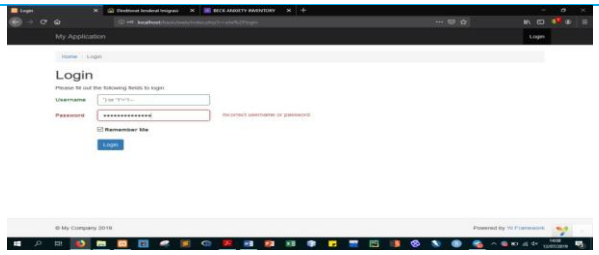


Figure.2 SQL Injection Prevention
Injection strings entered in the login fields failed. The Yii 2 framework correctly rejected all unauthorised inputs.

Table 6. Attack of SQL Injection

As illustrated in Figure. 1, the login form requires mandatory field completion, displaying warnings if any fields are left empty. Furthermore, as shown in Figure. 2, injection strings inserted into the login fields were rejected, and the application responded with an error message. The complete results of all eight SQL Injection test cases are summarised in Table 6, which confirms that none of the attack payloads succeeded in bypassing the login mechanism of the Yii 2 framework.

Cross-site Scripting (XSS). Three scenarios of script injection were attempted (Figure. 3–8), including cookie theft, forged POST actions, and phishing forms. As summarised in Table 7, all injections were blocked and returned "Page Not Found" or treated as plain text.
This outcome is explained by Yii 2's HTML encoding of outputs, which neutralises malicious scripts before rendering, and its CSRF token mechanism, which prevents forged requests. Developers only need to keep the default enableCsrfValidation = true configuration active for this protection to work.

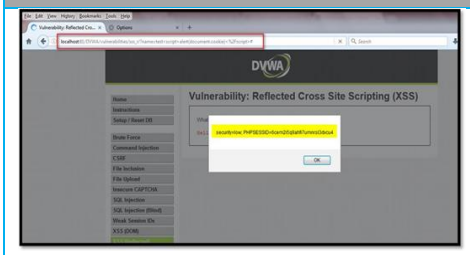
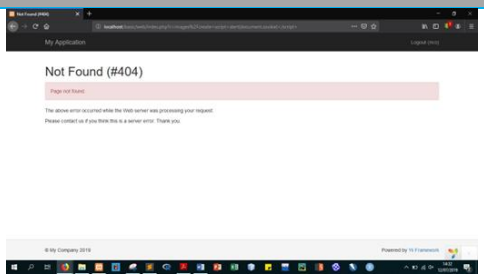
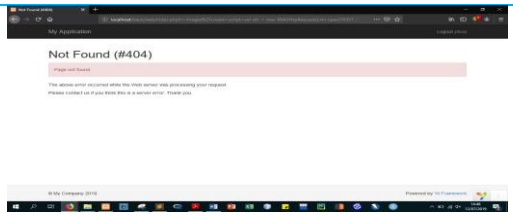
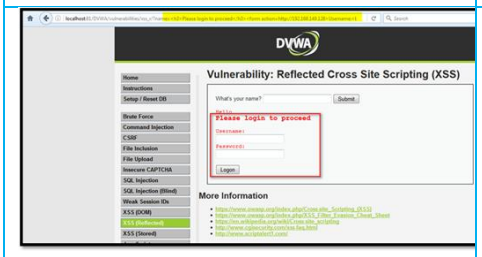
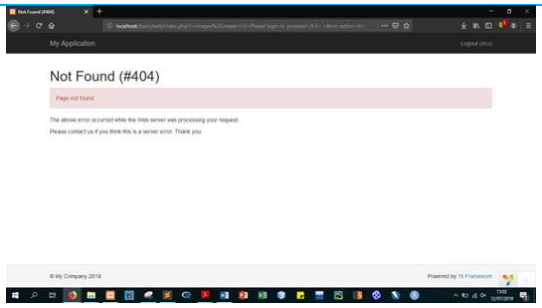
Scenario	Result
 Figure. 3 Session Hijacking via URL Script Injection The attacker injects JavaScript via the URL to steal cookies.	 Figure. 4 Session Hijacking via URL Script Injection The system returns a "Page Not Found" error. The script is not executed; it is treated as a plain URL. *(http://localhost/basic/aplikasi web/index.php?r=<script>new Image().src="http://localhost/basic/aplikasi web/index.php?r="+document.cookie;</script>)*
 Figure. 5 Perform Unauthorised Actions The attacker uses a script to forge a POST request and submit hidden form data.	 Figure. 6 Perform Unauthorised Actions Webpage not found. Script is blocked; the system does not respond to unauthorised POST actions. (http://localhost/basic/aplikasi web/index.php?r=images/create <script>var xhr=new XMLHttpRequest();xhr.open('POST','http://localhost /basic/aplikasi web/index.php?r=images/create',true); xhr.setRequestHeader('Content-type','application/x-www-form-urlencoded');xhr.send('txtName=xss&mtxMessage=xss&btnSign= Sign+Guestbook');</script>)
 Figure. 7 Phishing via Fake Login Form An attacker embeds a fake login form to steal credentials.	 Figure. 8 Phishing via Fake Login Form The fake form does not render. Web server responds with "Page Not Found". *(http://localhost/basic/aplikasi web/index.php?r=images/create Please login to proceed <form action=http://localhost>Username: <input type="username" name="username"> Password: <input type="password" name="password"> <input type="submit" value="Logon"></form>)*

Table 7. Attack of XSS

Remote Access Trojan (RAT). When attempting to upload disguised .php and .exe payloads (Figure. 9–11), the application displayed a warning message and blocked the submission (Table 8).

Yii 2 enforces file validation rules, particularly MIME-type filtering and extension checks, which reject non-image files. Unlike SQLi and XSS protection, this requires explicit configuration by developers, such as specifying allowed file extensions (.jpg, .png). Without this, file upload remains a potential risk.

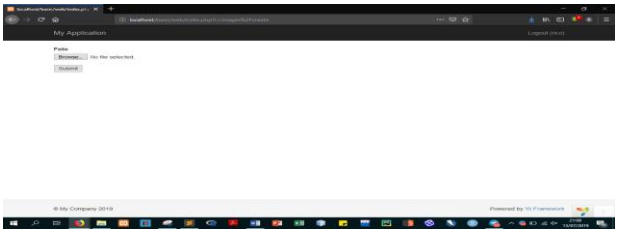


Figure. 9 Upload Form Interface
The user is presented with a file upload interface featuring a "Browse" and "Submit" button.

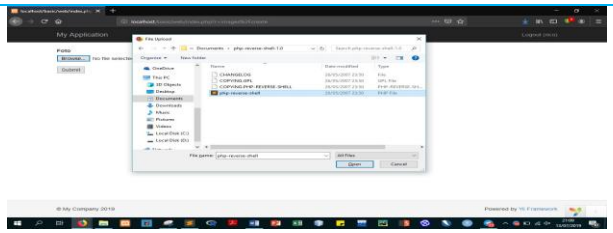


Figure. 10 Selection of Trojan File
The attacker selects a malicious .php or .exe file containing a Remote Access Trojan (RAT).

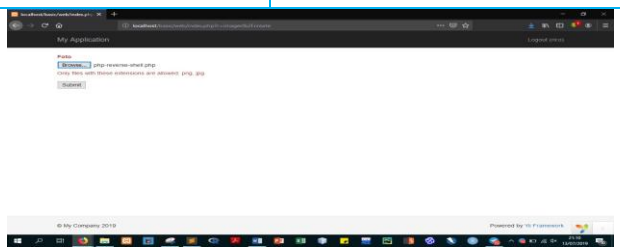


Figure. 11 Upload Attempt Fails
Upload blocked. Warning appears: "*Only .jpg and .png files are allowed.*" No execution or compromise occurs.

Table 8. Attack of RAT

Buffer Overflow. Login inputs ranging from 10 to 500 characters were tested (Figure. 12). As shown in Table 9, the framework rejected excessive inputs and returned only an "Incorrect username or password" warning.
This is the result of Yii 2's input length validation (maxlength rules) and internal handling of oversized strings, which prevent buffer memory corruption. While these safeguards are not typically aimed at preventing buffer overflows, they indirectly mitigate such risks.

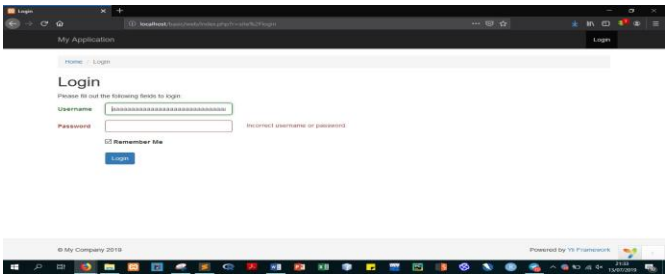


Figure. 12 Failed Buffer Overflow

All attempts failed. The application rejected input exceeding normal lengths. The system displayed only a warning message indicating an incorrect username or password.

Analysis of False Positives and Negatives. Across all experiments, no false negatives were observed—malicious payloads were consistently blocked. However, certain benign cases (e.g., overly long but harmless text) may be rejected, which could be considered false positives. Developers should refine validation rules to strike a balance between security and usability.

Overall, the experimental results indicate that Yii 2's built-in security mechanisms provide strong baseline protection against common web-based attacks. However, a deeper analysis reveals that this resilience is largely dependent on the framework's default configurations and

proper developer implementation. For instance, protections against SQL Injection and XSS are inherently robust due to automatic input parameterisation and output encoding, whereas file upload security relies more heavily on explicit developer-defined validation rules, such as restricting file types and MIME checking. This suggests that while the framework significantly reduces the likelihood of exploitation, it does not entirely eliminate risk, particularly in scenarios where security configurations are misapplied or omitted. Furthermore, the consistent classification of SQL Injection and Remote Attack as high-risk threats highlights their potential systemic impact, in contrast to XSS and Buffer Overflow, which tend to produce more localized or conditional effects. These findings emphasise that framework-level security must be complemented with secure coding practices and continuous testing to achieve comprehensive protection.

2. DREAD Risk Assessment

The DREAD model was used to quantify the risk level of each identified threat based on five dimensions: Damage Potential, Reproducibility, Exploitability, Affected Users, and Discoverability. The DREAD model was applied to quantify risk levels across five dimensions: Damage Potential, Reproducibility, Exploitability, Affected Users, and Discoverability. Scores ranged from 1–10 and were averaged for final classification.

- SQL Injection
  - Damage (9): May cause total loss of control over the database.
  - Reproducibility (6): Requires query knowledge but can be easily replicated.
  - Exploitability (8): Automated tools (e.g., sqlmap) facilitate exploitation.
  - Affected Users (9): Nearly all users are impacted if data is exposed.
  - Discoverability (6): Can be revealed through systematic input testing. → Average score 7.6, classified as *High*.
- Cross-site Scripting (XSS)
  - Damage (6): Enables theft of user data but rarely causes permanent server compromise.
  - Reproducibility (6): Easy to replicate if input validation is weak.
  - Exploitability (7): Simple scripts can be injected via web forms.
  - Affected Users (4): Limited to users who access compromised pages.
  - Discoverability (5): Typically found by attackers via input probing. → Average score 5.6, classified as *Medium*.
- Remote Attack
  - Damage (9): Grants attackers full control over the server.
  - Reproducibility (7): Payloads can be reused for repeated exploitation.
  - Exploitability (8): RAT tools make execution straightforward.
  - Affected Users (8): All users relying on the system are affected.
  - Discoverability (7): File upload vulnerabilities are relatively easy to test. → Average score 7.8, classified as *High*.
- Buffer Overflow
  - Damage (7): May cause crashes or arbitrary code execution.
  - Reproducibility (6): Requires carefully crafted input.
  - Exploitability (6): Less trivial than SQLi but feasible with the right payload.
  - Affected Users (7): Broadly impacts users if the system becomes unavailable.
  - Discoverability (5): Typically harder to detect without fuzzing. → Average score 6.2, classified as *Medium*.

Each threat was scored, and the average risk level was calculated as shown in Table 10.

Table 9. DREAD Scoring Results

Threat	D	R	E	A	D	Total	Avg. Score	Risk Level
SQL Injection	9	6	8	9	6	38	7.60	High
Cross-site Scripting	6	6	7	4	5	28	5.60	Medium
Remote Attack	9	7	8	8	7	39	7.80	High
Buffer Overflow	7	6	6	7	5	31	6.20	Medium

Based on the DREAD assessment, SQL Injection (score: 38, avg: 7.60) and Remote Attack (score: 39, avg: 7.80) were classified as High-risk threats, while Cross-site Scripting (score: 28, avg: 5.60) and Buffer Overflow (score: 31, avg: 6.20) were categorised as Medium-risk threats.

6. CONCLUSION

This study demonstrates that applying a Secure Software Development Lifecycle (SSDLC) approach, integrated with STRIDE and DREAD methodologies, is effective in assessing and mitigating threats in web-based applications. Four major security threats were tested on a Yii 2-based file uploader application: SQL Injection, Cross-site Scripting (XSS), Remote Attack (via RAT payload), and Buffer Overflow.

The simulated attack scenarios, involving complex payloads and input manipulations, revealed that the application was resilient to all tested exploits. SQL Injection and Remote Attack scored high on the DREAD risk model. At the same time, XSS and Buffer Overflow were categorised as medium risk, underscoring the importance of layered defences against high-severity threats.

The novelty of this research lies in being the first study to validate the effectiveness of Yii 2's built-in security features against file upload attacks using an integrated STRIDE–DREAD approach. This combined framework provides not only a classification of threats but also a quantification of risk severity, offering a structured basis for prioritising mitigations.

Overall, the findings confirm that Yii 2, when properly implemented, provides a robust baseline protection against common web threats. At the same time, the study emphasises the significance of early-stage threat modelling and structured risk assessment as proactive measures in secure software design, particularly for applications involving file handling and user authentication.

REFERENCES

[1] OWASP Foundation. "*File Upload Cheat Sheet*." *OWASP Cheat Sheet Series*, 2021cheatsheetseries.owasp.org. (Guidelines for secure file upload handling, covering extension allowlists, content validation, storage location, etc.)

[2] OWASP Foundation. "*Input Validation Cheat Sheet*." *OWASP Cheat Sheet Series*, 2021cheatsheetseries.owasp.org. (Best practices for validating and sanitizing user inputs, including file upload inputs and content checks.)

[3] Lee, T., Wi, S., Lee, S., & Son, S. (2020). "FUSE: Finding File Upload Bugs via Penetration Testing." In *Proc. Network and Distributed System Security Symposium (NDSS 2020)*ndss-symposium.org. (Introduces FUSE, a penetration testing tool that discovered unrestricted file upload (UFU) vulnerabilities in real-world PHP applications.)

[4] Wichmann, P., Groddeck, A., & Federrath, H. (2022). "FileUploadChecker: Detecting and Sanitizing Malicious File Uploads in Web Applications at the Request Level." In *Proc. 17th Intl. Conf. on Availability, Reliability and Security (ARES 2022)*dblp.org. (Proposes a server-side tool to automatically detect potentially malicious uploads and evaluate file upload security in popular CMS platforms.)

[5] Yenduri, R., & Al-khassaweneh, M. (2022). "PHP: Vulnerabilities and Solutions." In *Proc. 2nd Intl. Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC 2022)*, pp. 391–396mdpi.com. (Analyzes common PHP web application vulnerabilities—including unrestricted file uploads—and discusses prevention techniques within two custom PHP apps.)

[6] Neef, S., & Oudeh, M. (2024). "Bringing UFUs Back into the Air with FUEL: A Framework for Evaluating the Effectiveness of Unrestricted File Upload Vulnerability Scanners." In *Proc. 20th Intl. Conf. on Detection of Intrusions, Malware & Vulnerability Assessment (DIMVA 2024)*it-solutions-neef.de. (Presents the FUEL framework and benchmark for systematically assessing and comparing file upload vulnerability scanners using diverse UFU scenarios.)

[7] Bui, F. M., & Khondoker, R. (2024). "STRIDE-Based Cybersecurity Threat Modeling, Risk Assessment and Treatment of an In-Vehicle Infotainment System." *Vehicles*, 6(3), 1140–1163mdpi.com. (Demonstrates threat modeling with STRIDE and risk evaluation with DREAD, illustrating how to identify and mitigate security threats in complex systems – methodology applicable to secure software design and uploads.)

[8] NIST. (2022). *Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities*. NIST Special Publication 800-218nvlpubs.nist.gov. (Official guidelines for integrating security best practices into the Software Development Lifecycle (SSDLC), addressing areas like threat modeling, secure coding, and vulnerability mitigation processes.)

[9] Quyen, N. V. (2023). "Hands-on Training for Mitigating Web Application Vulnerabilities." Master's Thesis, Japan Advanced Institute of Science and Technology (JAIST). Available: http://hdl.handle.net/10119/18734